Matematica delle Reti Neurali

Volume III

Modelli generativi , Reinforcement Learning , Self-supervised Learning , Meta-learning , Teoria statistica , Sistemi dis

Silvano Salvador

Matematica delle Reti Neurali

Modelli generativi · Reinforcement Learning · Self-supervised Learning · Meta-learning · Teoria statistica · Sistemi distribuiti · Vision · Time series

Silvano Salvador · Continuazione dei Volumi I (Parti 0–50) e II (Parti 51–100)

Parti 101–144 Derivazioni complete RL da zero Teoria PAC rigorosa ZeRO e quantizzazione DETR · SAM · PatchTST

Introduzione al Volume III

Questo volume completa la trilogia trattando otto aree rimaste scoperte nei volumi precedenti.

Il Volume I ha costruito le fondamenta — dal neurone artificiale alla backpropagation tensoriale, dagli ottimizzatori alla normalizzazione, dalle CNN ai Transformer, dalla statistica bayesiana ai normalizing flows di base. Il Volume II ha approfondito le frontiere: architetture open-weight, modelli di diffusione con framework SDE, equivarianza di gruppo, Flow Matching, NTK, memoria associativa, alignment e federated learning. Il presente volume colma le lacune tematiche più rilevanti e aggiunge argomenti di nuova centralità.

Struttura del volume

Il testo è organizzato in otto sezioni tematiche (Parti 101–144):

- **Sezione A (§101–108):** Modelli generativi con derivazioni complete — VAE con ELBO e reparametrizzazione, GAN con analisi delle divergenze, DCGAN, WGAN-GP, StyleGAN, NICE, RealNVP, Glow multi-scala, DDPM con derivazione passo-passo dell'ELBO, DDIM deterministico, Consistency Models
- **Sezione B (§109–116):** Reinforcement Learning da zero — MDP formale, equazioni di Bellman, value/policy iteration, Q-learning tabellare con convergenza, DQN, REINFORCE con Policy Gradient Theorem, Actor-Critic, GAE, PPO con derivazione completa del clipped objective, SAC con soft Bellman, Model-Based RL
- **Sezione C (§117–122):** Self-Supervised e Contrastive Learning — InfoNCE e NT-Xent rigorosi, SimCLR, MoCo con momentum encoder, BYOL senza negativi, MAE, DINO con collapse prevention
- **Sezione D (§123–127):** Meta-Learning — setup episodico N-way K-shot, Prototypical

Networks, MAML con second-order gradients, Reptile, Meta-SGD

- **Sezione E (§128–133):** Teoria Statistica dell'Apprendimento — VC dimension con lemma di Sauer-Shelah, PAC learning, Rademacher complexity con bound tight, stabilità algoritmica, double descent rigoroso, benign overfitting
- **Sezione F (§134–137):** Training Distribuito e Inferenza Efficiente — data/model/pipeline parallelism, ZeRO stage 1/2/3 con analisi della memoria, quantizzazione PTQ/QAT/GPTQ, pruning strutturato e non strutturato, knowledge distillation
- **Sezione G (§138–140):** Computer Vision Avanzata — DETR con Hungarian matching e bipartite loss, SAM con prompt encoder e mask decoder, DeiT, BEiT, MAE-ViT
- **Sezione H (§141–144):** Time Series e Forecasting — TFT con gating gerarchico, N-BEATS con basis expansion, PatchTST con patch tokenization, iTransformer e Mamba per sequenze temporali

Riferimento ai Volumi precedenti: la notazione segue la Sezione 0 del Volume I. I rimandi espliciti indicano la sezione di origine con la formula §N del Vol. I o Vol. II.

Variational Autoencoder: ELBO, Reparametrizzazione e β-VAE

Derivazione rigorosa dell'Evidence Lower BOund, il trucco di reparametrizzazione per il gradiente stocastico, e le estensioni per il disentanglement delle rappresentazioni latenti.

101.1 Derivazione dell'ELBO

Un VAE modella la distribuzione dei dati $p(\mathbf{x})$ tramite una variabile latente continua $\mathbf{z}$, secondo la formula di marginalizzazione $p(\mathbf{x}) = \int p(\mathbf{x}|\mathbf{z})p(\mathbf{z})\,d\mathbf{z}$. Questa quantità è intrattabile perché richiederebbe integrare su tutti i possibili valori di $\mathbf{z}$ — uno spazio di dimensione elevata. Il VAE aggira questo ostacolo in modo elegante: invece di massimizzare direttamente la log-verosimiglianza, introduce una distribuzione variazionale $q_\phi(\mathbf{z}|\mathbf{x})$ che approssima la vera posteriore $p(\mathbf{z}|\mathbf{x})$, e ottimizza un lower bound alla log-evidence noto come ELBO.

La derivazione dell'ELBO parte da un'identità algebrica fondamentale che scompone la log-evidence in due termini: uno ottimizzabile e uno sempre non-negativo. Il punto chiave è moltiplicare e dividere per la distribuzione variazionale $q_\phi(\mathbf{z}|\mathbf{x})$ — un artificio che introduce la distribuzione dell'encoder senza alterare il valore dell'integrale:

$\log p(\mathbf{x}) = \log \int p(\mathbf{x}|\mathbf{z})p(\mathbf{z})\,d\mathbf{z}$ Definizione di marginale

$= \log \int q_\phi(\mathbf{z}|\mathbf{x}) \frac{p(\mathbf{x}|\mathbf{z})p(\mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x})}\,d\mathbf{z}$ Moltiplico e divido per $q_\phi(\mathbf{z}|\mathbf{x})$ — l'encoder variazionale

$\geq \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log \frac{p(\mathbf{x}|\mathbf{z})p(\mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x})}\right]$ Disuguaglianza di Jensen: $\log \mathbb{E}[X] \geq \mathbb{E}[\log X]$ per $X > 0$

$= \mathbb{E}_{q_\phi}[\log p(\mathbf{x}|\mathbf{z})] - D_{\mathrm{KL}}(q_\phi(\mathbf{z}|\mathbf{x}) \| p(\mathbf{z}))$ Separando il log: $\log p(\mathbf{x}|\mathbf{z}) + \log p(\mathbf{z}) - \log q_\phi = \log p(\mathbf{x}|\mathbf{z}) - \log(q_\phi/p)$

$$\mathcal{L}_{\mathrm{ELBO}}(\theta, \phi; \mathbf{x}) = \underbrace{\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x}|\mathbf{z})]}_{\text{ricostruzione}} - \underbrace{D_{\mathrm{KL}}(q_\phi(\mathbf{z}|\mathbf{x}) \| p(\mathbf{z}))}_{\text{regolarizzazione}} \leq \log p(\mathbf{x})$$

Il **gap** tra log-evidence e ELBO è esattamente la divergenza KL tra la posteriore variazionale e quella vera: $\log p(\mathbf{x}) - \mathcal{L} = D_{\mathrm{KL}}(q_\phi(\mathbf{z}|\mathbf{x}) \| p(\mathbf{z}|\mathbf{x})) \geq 0$. Poiché la KL divergence è sempre non-negativa, l'ELBO è effettivamente un lower bound alla log-evidence. Ne consegue che massimizzare l'ELBO ha un duplice effetto: da un lato aumenta la log-evidenza $\log p(\mathbf{x})$ (obiettivo generativo), dall'altro avvicina q_ϕ alla vera posteriore $p(\mathbf{z}|\mathbf{x})$ (obiettivo inferenziale). Questo legame è la ragione per cui il VAE unifica inference e generazione in un unico framework.

Forma chiusa della KL con gaussiane

Quando si sceglie la posteriore variazionale gaussiana diagonale $q_\phi(\mathbf{z}|\mathbf{x}) = \mathcal{N}(\boldsymbol{\mu}_\phi(\mathbf{x}), \mathrm{diag}(\boldsymbol{\sigma}^2_\phi(\mathbf{x})))$ con prior $p(\mathbf{z}) = \mathcal{N}(\mathbf{0}, \mathbf{I})$, la KL divergence si calcola in forma chiusa — una delle ragioni principali per cui questa parametrizzazione è universalmente adottata:

$$D_{\mathrm{KL}}(q \| p) = -\frac{1}{2} \sum_{j=1}^{d} \left(1 + \log \sigma_j^2 - \mu_j^2 - \sigma_j^2\right)$$

La derivazione per una singola dimensione procede calcolando esplicitamente l'integrale della KL: $D_{\mathrm{KL}}(\mathcal{N}(\mu, \sigma^2) \| \mathcal{N}(0,1)) = \frac{1}{2}(\mu^2 + \sigma^2 - 1 - \log \sigma^2)$. Poiché le dimensioni di $\mathbf{z}$ sono indipendenti sotto la parametrizzazione diagonale, la KL totale è semplicemente la somma su tutte le d dimensioni. Questo termine ha una bella interpretazione: penalizza encoder che producono medie $\boldsymbol{\mu}$ lontane dall'origine e varianze $\boldsymbol{\sigma}^2$ molto diverse da 1, spingendo la distribuzione latente verso il prior standard.

101.2 Il Trucco di Reparametrizzazione

Il termine di ricostruzione $\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x}|\mathbf{z})]$ richiede di differenziare rispetto a ϕ attraverso un'operazione di campionamento — operazione che non ammette gradiente diretto. Il gradiente $\nabla_\phi \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[f(\mathbf{z})]$ non si calcola direttamente poiché la distribuzione da cui si campiona dipende dai parametri ϕ che vogliamo ottimizzare. Il **reparametrization trick** risolve questo problema

con un'idea sorprendentemente semplice: si scrive $\mathbf{z}$ come funzione deterministica di un rumore ausiliario $\boldsymbol{\varepsilon}$ indipendente da ϕ:

$$\mathbf{z} = \boldsymbol{\mu}_\phi(\mathbf{x}) + \boldsymbol{\sigma}_\phi(\mathbf{x}) \odot \boldsymbol{\varepsilon}, \qquad \boldsymbol{\varepsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$$

Con questa riscrittura, il campionamento casuale è separato dalla dipendenza dai parametri: $\boldsymbol{\varepsilon}$ è fisso e ϕ appare solo nella trasformazione deterministica. Il gradiente scorre ora liberamente attraverso $\boldsymbol{\mu}_\phi$ e $\boldsymbol{\sigma}_\phi$ per backpropagation standard, rendendo l'intero VAE addestrabile con gradient descent:

$$\nabla_\phi \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[f(\mathbf{z})] = \nabla_\phi \mathbb{E}_{\boldsymbol{\varepsilon}\sim\mathcal{N}(\mathbf{0},\mathbf{I})}[f(\boldsymbol{\mu}_\phi + \boldsymbol{\sigma}_\phi \odot \boldsymbol{\varepsilon})] = \mathbb{E}_{\boldsymbol{\varepsilon}}[\nabla_\phi f(\boldsymbol{\mu}_\phi + \boldsymbol{\sigma}_\phi \odot \boldsymbol{\varepsilon})]$$

L'aspettativa nella loss può ora essere stimata con un singolo campione $\boldsymbol{\varepsilon}$ per esempio, producendo uno stimatore unbiased con varianza sufficientemente bassa da permettere l'addestramento efficiente con mini-batch. Il gradiente è computabile con qualsiasi framework di differenziazione automatica, rendendo il VAE un modello pratico nonostante la sua formulazione probabilistica sofisticata.

101.3 β-VAE e Disentanglement

Il β-VAE (Higgins et al., 2017) estende il VAE standard con una modifica apparentemente piccola ma dall'impatto profondo: si introduce un peso $\beta > 1$ sulla penalità KL, creando una tensione più forte tra ricostruzione fedele e regolarizzazione dello spazio latente:

$$\mathcal{L}_{\beta\text{-VAE}} = \mathbb{E}_{q_\phi}[\log p_\theta(\mathbf{x}|\mathbf{z})] - \beta\, D_{\mathrm{KL}}(q_\phi(\mathbf{z}|\mathbf{x}) \| p(\mathbf{z}))$$

Con $\beta > 1$, la pressione verso il prior gaussiano isotropico è maggiore di quanto necessario per una semplice approssimazione variazionale. Questa regolarizzazione forzata induce le dimensioni di $\mathbf{z}$ a diventare statisticamente indipendenti, e spinge ciascuna dimensione a specializzarsi nel controllo di un singolo fattore generativo separato — pose, illuminazione, colore, scala, identità. Il compromesso inevitabile è che la qualità della ricostruzione decresce al crescere di β: più si comprime l'informazione nello spazio latente, meno dettagli è possibile recuperare. La metrica **DCI** (Disentanglement, Completeness, Informativeness) fornisce una valutazione del

disentanglement indipendente dal modello, misurando quanta variazione in ogni fattore generativo è catturata da dimensioni latenti distinte.

GAN: Formulazione Minimax, Divergenze e Instabilità

102.1 Il Gioco Minimax

L'idea fondamentale delle Generative Adversarial Networks (Goodfellow et al., 2014) è radicalmente diversa da quella dei VAE: invece di massimizzare una verosimiglianza, si formula la generazione come un **gioco competitivo** tra due reti che si addestrano simultaneamente. Il generatore $G_\theta : \mathcal{Z} \to \mathcal{X}$ trasforma rumore casuale in campioni plausibili; il discriminatore $D_\phi : \mathcal{X} \to [0,1]$ cerca di distinguere i campioni reali da quelli generati. La value function che formalizza questo confronto è:

$$V(G,D) = \mathbb{E}_{\mathbf{x}\sim p_{\text{data}}}[\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z}\sim p(\mathbf{z})}[\log(1 - D(G(\mathbf{z})))]$$

$$\min_G \max_D V(G,D)$$

Per analizzare le proprietà teoriche di questo gioco, si cerca il discriminatore ottimale D_G^* a parità di generatore G fisso. Poiché la value function può essere scritta come un integrale, e l'integrando dipende da $D(\mathbf{x})$ indipendentemente per ogni punto $\mathbf{x}$, la massimizzazione rispetto a D avviene puntualmente — ogni posizione nello spazio dei dati contribuisce in modo indipendente:

Per ogni $\mathbf{x}$, massimizzare $p_{\text{data}}(\mathbf{x}) \log D(\mathbf{x}) + p_G(\mathbf{x}) \log(1 - D(\mathbf{x}))$ L'integrale è massimizzato quando ogni integrando è massimizzato

Derivando rispetto a D e ponendo uguale a zero: $\frac{p_{\text{data}}}{D} - \frac{p_G}{1-D} = 0$ Condizione del primo ordine

$D_G^*(\mathbf{x}) = \frac{p_{\text{data}}(\mathbf{x})}{p_{\text{data}}(\mathbf{x}) + p_G(\mathbf{x})}$ Risoluzione dell'equazione

102.2 Connessione con la Jensen-Shannon Divergence

Sostituendo D_G^* nella value function:

$V(G, D_G^*) = \mathbb{E}_{p_{\text{data}}}\left[\log \frac{p_{\text{data}}}{p_{\text{data}} + p_G}\right] + \mathbb{E}_{p_G}\left[\log \frac{p_G}{p_{\text{data}} + p_G}\right]$ Sostituzione di D_G^*

$= -\log 4 + D_{\mathrm{KL}}\left(p_{\mathrm{data}} \,\middle\|\, \frac{p_{\mathrm{data}}+p_G}{2}\right) + D_{\mathrm{KL}}\left(p_G \,\middle\|\, \frac{p_{\mathrm{data}}+p_G}{2}\right)$ Aggiungendo e sottraendo $\log 4$ e scrivendo in forma KL

$= -\log 4 + 2\,\mathrm{JSD}(p_{\mathrm{data}} \| p_G)$ Definizione di Jensen-Shannon Divergence: $\mathrm{JSD}(P\|Q) = \frac{1}{2}D_{\mathrm{KL}}(P\|\frac{P+Q}{2}) + \frac{1}{2}D_{\mathrm{KL}}(Q\|\frac{P+Q}{2})$

$$\min_G V(G, D_G^*) = -\log 4 + 2\,\mathrm{JSD}(p_{\mathrm{data}} \| p_G) \geq -\log 4$$

L'equilibrio ottimale del gioco è raggiunto quando $p_G = p_{\mathrm{data}}$, ovvero quando il generatore riproduce perfettamente la distribuzione dei dati reali. In questo punto la Jensen-Shannon Divergence si annulla ($\mathrm{JSD} = 0$) e il discriminatore ottimale diventa $D^* = 1/2$ ovunque — non riesce più a distinguere reale da sintetico con probabilità migliore del caso. Questo risultato teorico è elegante, ma come vedremo, raggiungerlo nella pratica è tutt'altro che banale.

102.3 Instabilità e Mode Collapse: Analisi

Il problema principale della formulazione originale emerge dalla geometria delle distribuzioni: $\mathrm{JSD}(P\|Q) = \log 2$ quando P e Q hanno supporti disgiunti — una condizione quasi sempre verificata nelle prime fasi del training, poiché p_G e p_{data} vivono in regioni diverse dello spazio dei dati ad alta dimensione. Quando il discriminatore è (quasi) perfetto, il gradiente che raggiunge il generatore è nullo o trascurabilmente piccolo: la rete generativa non riceve segnale utile per migliorare.

Il **mode collapse** è un'altra patologia caratteristica del training GAN: il generatore apprende a mappare molti vettori $\mathbf{z}$ diversi verso un sottoinsieme ristretto di $\mathcal{X}$, producendo una distribuzione generata con varianza quasi nulla concentrata attorno a pochi "mode" preferiti. Formalmente, p_G collassa su un supporto di misura zero rispetto a p_{data}. Dal punto di vista game-theoretic, il generatore ha trovato una strategia locally optimal contro il discriminatore corrente, ma globalmente subottimale: inganna il discriminatore con un insieme ristretto di campioni plausibili invece di esplorare l'intera distribuzione dei dati.

Cause principali di instabilità GAN

- **Oscillazioni:** G e D si inseguono senza convergere — il gioco non ha un punto sella stabile con SGD
- **Gradient saturation:** $\log(1 - D(G(\mathbf{z}))) \to 0$ quando D è troppo buono — si usa $-\log D(G(\mathbf{z}))$ come surrogate loss per G (non-saturating GAN)

- **Mode collapse:** G trova una singola immagine che inganna D, senza diversità
- **Training asimmetrico:** D più potente di $G \to$ gradiente nullo; G più potente $\to D$ non informativo

DCGAN, WGAN-GP e StyleGAN

103.1 DCGAN: Architettura e Linee Guida

DCGAN (Radford et al., 2016) rappresenta il primo tentativo sistematico di stabilizzare il training GAN attraverso scelte architetturali rigorose, piuttosto che modifiche alla funzione obiettivo. Prima di DCGAN, addestrare una GAN su immagini realistiche era considerato estremamente difficile e richiedeva esperienza empirica considerevole. Il contributo principale è un insieme di best practice architetturali, ciascuna motivata da ragioni teoriche o empiriche precise:

Componente	Scelta DCGAN	Motivazione
Pooling	Strided convolution (D), fractional-strided conv (G)	Il downsampling/upsampli
Normalizzazione	Batch Normalization (tranne primo layer G e ultimo D)	Stabilizza il flusso del grad
Attivazione G	ReLU in tutti i layer, Tanh all'output	Output in $[-1, 1]$, compatil
Attivazione D	Leaky ReLU ($\alpha = 0.2$)	Evita sparse gradients
Input G	$\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, $d_z = 100$	Rumore gaussiano, nessun

L'architettura completa del generatore per immagini 64×64 segue una struttura piramidale di upsampling progressivo: $\mathbf{z} \in \mathbb{R}^{100} \to \text{ConvT}(512, 4, 1) \to \text{ConvT}(256, 4, 2) \to \text{ConvT}(128, 4, 2) \to \text{ConvT}(64, 4, 2) \to \text{ConvT}(3, 4, 2, \tanh)$. Ogni blocco raddoppia la risoluzione spaziale dimezzando i canali, mantenendo costante il volume computazionale tra i layer. Questa progressione dalla rappresentazione semantica ad alta dimensione (4×4×512) all'immagine a bassa dimensione (64×64×3) rispecchia il processo inverso di un encoder convolutivo.

103.2 WGAN-GP: Derivazione Completa del Gradient Penalty

WGAN (Arjovsky et al., 2017) affronta il problema del gradiente nullo alla radice, sostituendo la Jensen-Shannon Divergence con la Wasserstein-1 distance — una metrica che fornisce un segnale di gradiente utile anche quando le distribuzioni hanno supporti disgiunti (si veda la dualità Kantorovich-Rubinstein, §61.2 del Vol. II). Il discriminatore di WGAN non è più un classificatore binario ma un *critic* a valori reali che deve essere una funzione 1-Lipschitz. La versione originale di WGAN implementa questo vincolo con il weight clipping, ma questa scelta

porta a gradienti che esplodono o svaniscono. WGAN-GP (Gulrajani et al., 2017) risolve il problema con una penalità sul gradiente più sofisticata:

$$\mathcal{L}_D = \underbrace{\mathbb{E}_{\mathbf{x}\sim p_{\text{data}}}[D(\mathbf{x})] - \mathbb{E}_{\tilde{\mathbf{x}}\sim p_G}[D(\tilde{\mathbf{x}})]}_{\text{approssimazione di } -W_1} + \underbrace{\lambda\, \mathbb{E}_{\hat{\mathbf{x}}\sim p_{\hat{\mathbf{x}}}}\left[(\|\nabla_{\hat{\mathbf{x}}} D(\hat{\mathbf{x}})\|_2 - 1)^2\right]}_{\text{gradient penalty}}$$

I punti interpolati $\hat{\mathbf{x}} = t\mathbf{x} + (1-t)\tilde{\mathbf{x}}$ con $t \sim \mathcal{U}[0,1]$ giacciono sulle linee rette tra campioni reali e generati — le geodetiche approssimate nello spazio dei dati. La penalità impone $\|\nabla D\|_2 = 1$ lungo queste geodetiche, garantendo che il critic soddisfi la condizione di Lipschitz esattamente dove conta per il trasporto ottimale. Questa condizione è sia necessaria che sufficiente per la 1-Lipschitzianità sul supporto congiunto delle due distribuzioni, come dimostrato tramite la teoria del complementary slackness della programmazione lineare continua. In pratica, $\lambda = 10$ è il valore standard per il peso della penalità.

La loss del generatore è semplicemente: $\mathcal{L}_G = -\mathbb{E}_{\mathbf{z}}[D(G(\mathbf{z}))]$ — il generatore cerca di massimizzare il valore che il critic assegna ai propri campioni, spingendo p_G verso le regioni ad alta probabilità sotto p_{data}. A differenza della formulazione originale, questo obiettivo fornisce un gradiente stabile in ogni fase del training, poiché la Wasserstein distance è continua e differenziabile anche quando le distribuzioni hanno supporti disgiunti.

103.3 StyleGAN: Mapping Network e AdaIN

StyleGAN (Karras et al., 2019) introduce un'innovazione architetturale fondamentale: la separazione tra lo spazio del rumore $\mathcal{Z}$ (gaussiano isotropico) e lo spazio degli stili $\mathcal{W}$ (più strutturato). Il **mapping network** $f : \mathcal{Z} \to \mathcal{W}$ — un MLP con 8 layer — trasforma il rumore $\mathbf{z}$ in un vettore di stile $\mathbf{w}$ che ha una geometria più regolare e disentangled. Questo $\mathbf{w}$ controlla poi la sintesi dell'immagine ad ogni risoluzione separatamente, tramite Adaptive Instance Normalization (AdaIN):

$$\text{AdaIN}(\mathbf{x}_i, \mathbf{y}) = \mathbf{y}_{s,i} \frac{\mathbf{x}_i - \mu(\mathbf{x}_i)}{\sigma(\mathbf{x}_i)} + \mathbf{y}_{b,i}$$

dove $(\mathbf{y}_s, \mathbf{y}_b) = \text{A}(\mathbf{w})$ sono scala e bias specifici per il layer, appresi tramite una rete di affine transformation A. L'AdaIN normalizza prima il feature map (rimuovendo le informazioni di stile) e poi ne inietta di nuove tramite $\mathbf{w}$.

Il rumore stocastico per i dettagli (texture, capelli) è aggiunto direttamente al feature map

prima di ogni AdaIN — separando controllo dello stile (deterministico da $\mathbf{w}$) da variazione stocastica fine (dal rumore). StyleGAN2 rimuove l'AdaIN e usa la **weight demodulation**: i pesi del layer conv sono scalati da $\mathbf{w}$ e poi normalizzati per preservare la varianza dell'output.

NICE e RealNVP: Coupling Layers Rigorosi

104.1 NICE: Non-linear Independent Components Estimation

NICE (Dinh et al., 2015) — acronimo di Non-linear Independent Components Estimation — è il primo modello della famiglia dei **normalizing flow** basato su un'idea architettonica ingegnosa: costruire trasformazioni invertibili con Jacobiano calcolabile in modo efficiente, aggirando così il costo proibitivo della fattorizzazione LU. L'intuizione è semplice: se si partiziona il vettore $\mathbf{x} = (x_{1:k}, x_{k+1:d})$ in due blocchi e si trasforma solo il secondo condizionatamente al primo, la struttura triangolare del Jacobiano rende il determinante banale da calcolare:

$$y_{1:k} = x_{1:k} \quad \text{(identità)}$$

$$y_{k+1:d} = x_{k+1:d} + m_\theta(x_{1:k}) \quad \text{(addizione)}$$

dove $m_\theta : \mathbb{R}^k \to \mathbb{R}^{d-k}$ è una rete neurale arbitraria — la sua complessità non influisce sulla invertibilità né sul calcolo del determinante. L'inversione è banale proprio perché la prima metà rimane invariata: $x_{1:k} = y_{1:k}$ e $x_{k+1:d} = y_{k+1:d} - m_\theta(y_{1:k})$. Questo schema permette di usare reti profonde e arbitrariamente espressive per m_θ senza compromettere le proprietà di invertibilità. Il Jacobiano della trasformazione ha una struttura triangolare a blocchi:

$$\mathbf{J} = \frac{\partial \mathbf{y}}{\partial \mathbf{x}} = \begin{pmatrix} \mathbf{I}_k & \mathbf{0} \\ \frac{\partial m_\theta}{\partial x_{1:k}} & \mathbf{I}_{d-k} \end{pmatrix}$$

Questa è una matrice triangolare inferiore a blocchi, il cui determinante è semplicemente il prodotto dei blocchi diagonali: $|\det \mathbf{J}| = |\det \mathbf{I}_k| \cdot |\det \mathbf{I}_{d-k}| = 1$. Ne consegue che la log-likelihood si calcola esattamente e senza alcun costo aggiuntivo per il Jacobiano. Per aumentare l'espressività del flusso, si alternano due diverse partizioni (es. dimensioni pari e dispari) e si applicano L coupling layers in composizione. Poiché ogni layer ha $\det = 1$, la log-likelihood

totale è semplicemente $\log p(\mathbf{x}) = \log p(f(\mathbf{x}))$ — la distribuzione dei dati trasformati sotto il prior.

L'ultimo layer di NICE è una rescaling layer diagonale $y_i = e^{s_i} x_i$, con s_i parametri appresi, che contribuisce al determinante con $\log|\det| = \sum_i s_i$. Questo strato rompe la simmetria della trasformazione precedente ($\det = 1$ per tutti i coupling layers) e permette al modello di adattare la varianza marginale di ogni dimensione latente alla distribuzione reale dei dati — una forma di auto-adattamento che migliora significativamente la qualità generativa.

104.2 RealNVP: Affine Coupling Layers

RealNVP (Dinh et al., 2017) estende NICE sostituendo il semplice accoppiamento additivo con un **affine coupling** più espressivo, che combina scala e traslazione dipendenti dall'input. Questa generalizzazione mantiene tutta la trattabilità di NICE ma permette al flusso di deformare lo spazio latente in modo molto più ricco:

$$y_{1:k} = x_{1:k}$$

$$y_{k+1:d} = x_{k+1:d} \odot \exp(s_\theta(x_{1:k})) + t_\theta(x_{1:k})$$

dove $(s_\theta, t_\theta) : \mathbb{R}^k \to \mathbb{R}^{d-k} \times \mathbb{R}^{d-k}$ sono le reti che producono rispettivamente la scala e la traslazione, condizionate alla prima metà dell'input. L'inversione rimane esplicita e computazionalmente economica: $x_{k+1:d} = (y_{k+1:d} - t_\theta(y_{1:k})) \odot \exp(-s_\theta(y_{1:k}))$. Il Jacobiano è triangolare a blocchi con il blocco $(k+1:d, k+1:d)$ diagonale, dove i coefficienti diagonali sono esattamente i fattori di scala $\exp(s_{\theta,i})$:

$$\log|\det \mathbf{J}| = \sum_{i=k+1}^{d} s_{\theta,i}(x_{1:k})$$

Il costo del determinante è $O(d)$ invece di $O(d^3)$ che richiederebbe la fattorizzazione LU di un Jacobiano arbitrario — un risparmio computazionale enorme che rende i normalizing flow praticamente scalabili. RealNVP introduce inoltre una **multi-scale architecture** che si ispira alla struttura wavelet: dopo ogni blocco di coupling layers, il tensore viene diviso a metà lungo

i canali. Una metà viene "emessa" direttamente e modellata come rumore gaussiano (bypassa il resto della rete), mentre l'altra metà continua ad essere trasformata dai layer successivi. Questa strategia riduce progressivamente la dimensionalità, abbassa il costo computazionale e permette al modello di catturare strutture a scale diverse.

Glow: Architettura Multi-Scala Completa

105.1 Struttura Gerarchica

Glow (Kingma e Dhariwal, 2018) rappresenta l'apice dei normalizing flow convoluzionali, combinando RealNVP con due innovazioni fondamentali: le 1×1 convoluzioni invertibili come alternativa all'alternanza fissa delle partizioni, e l'ActNorm come sostituto del batch normalization. Il modello è organizzato gerarchicamente in L livelli, ciascuno composto da K step indipendenti. Ogni step è la composizione di tre trasformazioni invertibili in sequenza:

$$\mathbf{h} \xrightarrow{\text{ActNorm}} \mathbf{h}' \xrightarrow{\text{Conv } 1\times1} \mathbf{h}'' \xrightarrow{\text{Affine Coupling}} \mathbf{h}'''$$

Ad ogni livello $\ell < L$ si applica una operazione di **squeeze** che riduce la risoluzione spaziale dimezzando altezza e larghezza mentre quadruplica i canali: $(H/2^\ell, W/2^\ell, C \cdot 4^\ell)$. Seguono poi i K step del livello, dopo i quali si effettua uno **split**: metà dei canali vengono proiettati direttamente sulla distribuzione prior gaussiana e "emessi" dal modello, mentre la restante metà prosegue al livello successivo per subire ulteriori trasformazioni. Questo schema introduce correlazioni condizionate tra scale diverse e fa sì che il modello impari prima le strutture a bassa frequenza e poi quelle ad alta frequenza — analogamente alla codifica multi-scala in wavelet.

Log-likelihood completa

Per un'immagine discreta $\mathbf{x} \in \mathbb{R}^{H\times W\times C}$ con valori interi in $\{0,\dots,255\}$, una modellazione diretta della distribuzione continua causerebbe degenerazioni (massa concentrata sui valori interi). Si aggiunge quindi un rumore di dequantizzazione uniforme $\mathbf{u} \sim \mathcal{U}[0,1)^d$ e si modella la distribuzione di $\mathbf{x} + \mathbf{u}$ — tecnica che trasforma il problema da discreto a continuo senza distorcere la distribuzione:

$$\log p(\mathbf{x}) \approx \log p_Z(f(\mathbf{x})) + \sum_{\ell=1}^{L}\sum_{k=1}^{K} \log\left|\det \mathbf{J}_{\ell,k}\right| - d\log 256$$

dove $d = H \cdot W \cdot C$ è la dimensionalità totale dell'immagine. Il termine $-d \log 256$ normalizza per la discretizzazione degli pixel. Il contributo al log-determinante di ogni step si scompone in tre termini indipendenti: $\log|\det| = \sum_i \text{ActNorm}_i + H \cdot W \log|\det \mathbf{W}_\ell| + \sum_i \text{affine}_i$. Notare che il contributo della convoluzione 1×1 scala con il numero di posizioni spaziali $H \cdot W$: ogni posizione applica la stessa trasformazione $\mathbf{W}_\ell$, e il log-determinante totale è la somma sui $H \cdot W$ siti.

DDPM: Derivazione Completa dell'Evidence Lower Bound della Diffusione

106.1 Il Processo Forward: Diffusione del Segnale

I Denoising Diffusion Probabilistic Models (DDPM, Ho et al., 2020) sono ispirati dalla termodinamica statistica: così come un sistema fisico raggiunge l'equilibrio termico attraverso l'aggiunta graduale di rumore, il modello definisce un processo di Markov forward che trasforma un'immagine reale $\mathbf{x}_0 \sim q(\mathbf{x}_0)$ in rumore gaussiano puro attraverso T passi incrementali. L'idea centrale è poi invertire questo processo — imparando a rimuovere il rumore passo dopo passo — per generare nuove immagini:

$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t\mathbf{I})$$

Il noise schedule $\beta_1 < \beta_2 < \cdots < \beta_T$ controlla la velocità con cui il segnale viene corrotto: valori piccoli nelle prime fasi preservano la struttura, valori più grandi verso la fine distruggono i dettagli residui. La proprietà fondamentale che rende DDPM pratico è che il processo forward è campionabile in un **singolo passo** senza dover simulare l'intera catena Markoviana, grazie alla stabilità per composizione delle gaussiane. Definendo $\alpha_t = 1 - \beta_t$ e il prodotto cumulato $\bar{\alpha}_t = \prod_{s=1}^{t} \alpha_s$:

$$q(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t; \sqrt{\bar{\alpha}_t}\mathbf{x}_0, (1-\bar{\alpha}_t)\mathbf{I})$$

$$\mathbf{x}_t = \sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\varepsilon}, \quad \boldsymbol{\varepsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$$

La derivazione procede per sostituzione ricorsiva: $\mathbf{x}_t = \sqrt{\alpha_t}\mathbf{x}_{t-1} + \sqrt{\beta_t}\boldsymbol{\varepsilon}_{t-1} = \sqrt{\alpha_t\alpha_{t-1}}\mathbf{x}_{t-2} + \sqrt{\alpha_t\beta_{t-1}}\boldsymbol{\varepsilon}_{t-2} + \sqrt{\beta_t}\boldsymbol{\varepsilon}_{t-1}$. Poiché la somma di variabili gaussiane indipendenti è gaussiana

con varianza sommata, tutti i termini di rumore si combinano in un unico termine: $\mathbf{x}_t = \sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\varepsilon}$, dove $\boldsymbol{\varepsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. Questa formula è cruciale: permette di campionare $\mathbf{x}_t$ direttamente da $\mathbf{x}_0$ a qualsiasi timestep, senza simulare i passi intermedi.

106.2 Il Processo Reverse: Denoising

Il processo inverso ideale richiederebbe di campionare da $q(\mathbf{x}_{t-1}|\mathbf{x}_t)$, che dipende dall'intera distribuzione dei dati ed è intrattabile. Tuttavia, *condizionata* su $\mathbf{x}_0$, la reverse posterior $q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)$ è gaussiana e ha una forma chiusa. Si ottiene applicando il teorema di Bayes e il fatto che $\mathbf{x}_{t-1}$ e $\mathbf{x}_t$ sono correlati attraverso $\mathbf{x}_0$:

$$q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) = \mathcal{N}(\mathbf{x}_{t-1}; \tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t, \mathbf{x}_0), \tilde{\beta}_t \mathbf{I})$$

La media e varianza si derivano completando il quadrato nell'esponente della gaussiana prodotto $q(\mathbf{x}_t|\mathbf{x}_{t-1}) \cdot q(\mathbf{x}_{t-1}|\mathbf{x}_0)/q(\mathbf{x}_t|\mathbf{x}_0)$ — un calcolo algebrico che rivela la struttura gaussiana della posteriore condizionata. Il risultato è:

$$\tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t, \mathbf{x}_0) = \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}\mathbf{x}_0 + \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\mathbf{x}_t$$

$$\tilde{\beta}_t = \frac{(1-\bar{\alpha}_{t-1})\beta_t}{1-\bar{\alpha}_t}$$

106.3 ELBO della Diffusione: Decomposizione Completa

Il modello neurale apprende $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \boldsymbol{\mu}_\theta(\mathbf{x}_t, t), \Sigma_\theta(\mathbf{x}_t, t))$ — una rete che, dato un'immagine rumorosa al passo t, predice la distribuzione dell'immagine al passo $t-1$. L'addestramento massimizza la log-verosimiglianza tramite la stessa identità ELBO vista per i VAE, stavolta applicata alla catena di Markov reverse:

$$\log p(\mathbf{x}_0) \geq \mathbb{E}_q\left[\log \frac{p_\theta(\mathbf{x}_{0:T})}{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}\right] = -\mathcal{L}_{\text{VLB}}$$

Espandendo l'ELBO tramite la struttura Markoviana del processo (forward e reverse), si ottiene una decomposizione in tre tipi di termini con interpretazioni precise:

$$\mathcal{L}_{\text{VLB}} = \underbrace{D_{\text{KL}}(q(\mathbf{x}_T|\mathbf{x}_0)\|p(\mathbf{x}_T))}_{\mathcal{L}_T \approx 0} + \sum_{t=2}^{T} \underbrace{D_{\text{KL}}(q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)\|p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t))}_{\mathcal{L}_{t-1}} - \underbrace{\log p_\theta(\mathbf{x}_0|\mathbf{x}_1)}_{\mathcal{L}_0}$$

Ogni $\mathcal{L}_{t-1}$ è la KL tra due gaussiane: $D_{\text{KL}}(\mathcal{N}(\tilde{\boldsymbol{\mu}}_t, \tilde{\beta}_t\mathbf{I})\|\mathcal{N}(\boldsymbol{\mu}_\theta, \Sigma_\theta)) = \frac{1}{2\Sigma_\theta}\|\tilde{\boldsymbol{\mu}}_t - \boldsymbol{\mu}_\theta\|^2 + \text{const.}$

Sostituendo $\mathbf{x}_0 = (\mathbf{x}_t - \sqrt{1-\bar{\alpha}_t}\boldsymbol{\varepsilon})/\sqrt{\bar{\alpha}_t}$ nella formula di $\tilde{\boldsymbol{\mu}}_t$:

$$\tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t, \boldsymbol{\varepsilon}) = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\varepsilon}\right)$$

Se $\boldsymbol{\mu}_\theta(\mathbf{x}_t, t) = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\varepsilon}_\theta(\mathbf{x}_t, t)\right)$, il termine $\mathcal{L}_{t-1}$ diventa:

$$\mathcal{L}_{t-1} \propto \left\|\boldsymbol{\varepsilon} - \boldsymbol{\varepsilon}_\theta\left(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\varepsilon},\, t\right)\right\|^2$$

L'obiettivo pratico di DDPM: la rete $\boldsymbol{\varepsilon}_\theta$ predice il rumore aggiunto, e si campiona uniformemente $t \in \{1, \ldots, T\}$ e $\boldsymbol{\varepsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$.

DDIM: Sampling Deterministico e ODE View

107.1 Dalla Catena di Markov all'ODE

Una delle intuizioni più eleganti nella teoria della diffusione è quella alla base di DDIM (Song et al., 2021): l'addestramento di DDPM dipende esclusivamente dalle marginali $q(\mathbf{x}_t|\mathbf{x}_0)$, e non dalla struttura Markoviana del processo forward. Ciò significa che esiste un'intera famiglia di processi non-Markoviani che condividono le stesse marginali — e quindi lo stesso identico obiettivo di addestramento — ma che ammettono processi inversi con comportamento radicalmente diverso. La reverse transition generalizzata a questa famiglia è:

$$q_\sigma(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) = \mathcal{N}\left(\sqrt{\bar{\alpha}_{t-1}}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_{t-1}-\sigma_t^2}\cdot\frac{\mathbf{x}_t - \sqrt{\bar{\alpha}_t}\mathbf{x}_0}{\sqrt{1-\bar{\alpha}_t}},\, \sigma_t^2\mathbf{I}\right)$$

Il parametro σ_t governa la stocasticità del processo inverso: per $\sigma_t = \tilde{\beta}_t^{1/2}$ si recupera esattamente DDPM con la sua stocasticità originale. Per $\sigma_t = 0$ per tutti i t, il processo inverso diventa completamente **deterministico**: dato $\mathbf{x}_T$, l'intera traiettoria verso $\mathbf{x}_0$ è determinata in modo univoco. Questa è la scelta chiave di DDIM, e la formula di sampling corrispondente diventa:

$$\mathbf{x}_{t-1} = \sqrt{\bar{\alpha}_{t-1}} \underbrace{\frac{\mathbf{x}_t - \sqrt{1-\bar{\alpha}_t}\boldsymbol{\varepsilon}_\theta(\mathbf{x}_t, t)}{\sqrt{\bar{\alpha}_t}}}_{\text{predizione di } \mathbf{x}_0} + \sqrt{1-\bar{\alpha}_{t-1}} \cdot \boldsymbol{\varepsilon}_\theta(\mathbf{x}_t, t)$$

La natura deterministica di DDIM permette di saltare la maggior parte dei timestep: si usa solo un sottoinsieme $\{\tau_1, \ldots, \tau_S\} \subset \{1, \ldots, T\}$ con $S \ll T$, ottenendo un'accelerazione di 10–50× rispetto a DDPM senza ri-training della rete. Con $T = 1000$ timestep e $S = 50$, la qualità rimane comparabile mentre il costo computazionale si riduce di 20×. La corrispondente ODE continua nel limite $S \to \infty$ è:

$$d\mathbf{x} = \left[\frac{\mathbf{x} - \sqrt{1-\bar{\alpha}_t}\boldsymbol{\varepsilon}_\theta(\mathbf{x}, t)}{2\bar{\alpha}_t}\right] \frac{d\bar{\alpha}_t}{\bar{\alpha}_t}$$

Questa è precisamente la Probability Flow ODE di Song et al. (Parte 76 del Vol. II), riscritta nei termini dello schedule $\bar{\alpha}_t$ usato da DDPM. DDIM ne è la discretizzazione numerica con passo adattivo — si selezionano i timestep τ_i per minimizzare l'errore di discretizzazione dell'ODE, bilanciando velocità e qualità in modo controllato.

Consistency Models: Distillazione Progressiva

108.1 La Consistency Function

I Consistency Models (Song et al., 2023) pongono una domanda radicale: è possibile imparare a generare immagini di alta qualità con un *singolo* passo di rete neurale? La risposta è affermativa, a patto di adottare un cambio di prospettiva fondamentale. Invece di simulare l'ODE passo per passo, si definisce direttamente una funzione $f_\theta : \mathbb{R}^d \times [t_\epsilon, T] \to \mathbb{R}^d$ che mappa qualsiasi punto della traiettoria ODE — indipendentemente dal timestep — direttamente all'immagine originale $\mathbf{x}_0$. Una volta appresa tale funzione, la generazione riduce a un singolo forward pass:

$$f_\theta(\mathbf{x}_t, t) \approx \mathbf{x}_0 \quad \text{per tutti i punti } (\mathbf{x}_t, t) \text{ sulla traiettoria ODE}$$

La proprietà fondamentale che questa funzione deve soddisfare è la **consistency condition**: $f_\theta(\mathbf{x}_t, t) = f_\theta(\mathbf{x}_{t'}, t')$ per qualsiasi coppia di timestep t, t' che giace sulla stessa traiettoria ODE. Questa condizione garantisce che, partendo da qualsiasi punto lungo la traiettoria, si arrivi sempre alla stessa immagine originale. La parametrizzazione standard usa una skip connection per garantire la boundary condition in modo automatico:

$$f_\theta(\mathbf{x}, t) = c_{\text{skip}}(t)\mathbf{x} + c_{\text{out}}(t)F_\theta(\mathbf{x}, t)$$

I coefficienti c_{skip} e c_{out} sono scelti tali che $c_{\text{skip}}(t_\epsilon) = 1$ e $c_{\text{out}}(t_\epsilon) = 0$: in questo modo la boundary condition $f_\theta(\mathbf{x}_\epsilon, t_\epsilon) = \mathbf{x}_\epsilon$ è soddisfatta esattamente per costruzione, senza doverla imporre esplicitamente durante il training. Il modello deve quindi imparare solo le deviazioni da questa identità per i timestep $t > t_\epsilon$.

La modalità di addestramento per distillazione (**Consistency Distillation**) sfrutta un modello di diffusione già addestrato come teacher: il suo ODE solver genera coppie di punti adiacenti sulla stessa traiettoria, e la rete student impara a mapparli allo stesso $\mathbf{x}_0$:

$$\mathcal{L}_{\text{CD}} = \mathbb{E}\Big[d\Big(f_\theta(\mathbf{x}_{t_{n+1}}, t_{n+1}),\ f_{\theta^-}(\hat{\mathbf{x}}^\phi_{t_n}, t_n)\Big)\Big]$$

dove $\hat{\mathbf{x}}^\phi_{t_n}$ è un passo ODE da $\mathbf{x}_{t_{n+1}}$ usando il teacher, θ^- è un EMA di θ, e $d(\cdot,\cdot)$ è una metrica (LPIPS o ℓ_2). I consistency models permettono generazione in **1 passo** con qualità comparabile a DDPM con 1000 passi.

Analisi Comparativa: Qualità, Diversità e Tradeoff nei Modelli Generativi

108b.1 Metriche di Valutazione

La valutazione quantitativa dei modelli generativi si basa su tre dimensioni: **qualità** (realismo dei singoli campioni), **diversità** (copertura della distribuzione) e **fidelity** (corrispondenza con la distribuzione target).

Fréchet Inception Distance (FID)

FID (Heusel et al., 2017) misura la distanza tra la distribuzione delle feature di Inception di immagini reali e generate. Le feature estratte sono approssimate come gaussiane multivariata $(\boldsymbol{\mu}_r, \mathbf{C}_r)$ e $(\boldsymbol{\mu}_g, \mathbf{C}_g)$:

$$\text{FID} = \|\boldsymbol{\mu}_r - \boldsymbol{\mu}_g\|^2 + \text{tr}\big(\mathbf{C}_r + \mathbf{C}_g - 2(\mathbf{C}_r\mathbf{C}_g)^{1/2}\big)$$

Il termine $\text{tr}(\mathbf{C}_r + \mathbf{C}_g - 2(\mathbf{C}_r\mathbf{C}_g)^{1/2})$ è la distanza di Fréchet (Wasserstein-2) tra due gaussiane — si riduce a $\sum_i(\sqrt{\lambda_i^r} - \sqrt{\lambda_i^g})^2$ dove λ_i sono gli autovalori del prodotto delle covarianze. FID basso = distribuzione generata vicina a quella reale. Richiede 50k campioni per stime stabili.

Precision e Recall per Modelli Generativi

Kynkäänniemi et al. (2019) decompongono FID in precision (qualità) e recall (copertura) separati. Dati embedding $\phi(\mathbf{x})$ per immagini reali $\{r_i\}$ e generate $\{g_j\}$, il manifold reale $\mathcal{M}_r$ è il insieme dei punti entro raggio k-NN:

$$\text{Precision} = \frac{1}{|G|}\sum_j \mathbf{1}[\phi(g_j) \in \mathcal{M}_r], \qquad \text{Recall} = \frac{1}{|R|}\sum_i \mathbf{1}[\phi(r_i) \in \mathcal{M}_g]$$

Mode collapse → alta precision, basso recall. Mode dropping → alta precision, recall intermedio. Una distribuzione troppo diffusa → bassa precision, alto recall.

Tabella comparativa dei modelli generativi

Modello	**Training**	**Sampling**	**Log-lik. esatta**	**FID tipico (ImageNe**
GAN (StyleGAN3)	Instabile (minimax)	Rapido (1 step)	No	∼3–5
VAE	Stabile (ELBO)	Rapido	Lower bound	∼30–50
NF (Glow)	Stabile (MLE)	Rapido	Sì (esatta)	∼35
DDPM	Stabile	Lento (T step)	Lower bound	∼3–7
DDIM (S step)	Come DDPM	$S \ll T$ step	No	∼4–8
Consistency Model	Distillazione	1–2 step	No	∼4–6

108b.2 Conditional Generation e Classifier-Free Guidance

Il Classifier-Free Guidance (Ho e Salimans, 2022) permette di controllare la generazione condizionata senza un classificatore esterno. Si addestra un unico modello sia condizionato $\boldsymbol{\varepsilon}_\theta(\mathbf{x}_t, t, \mathbf{c})$ che non condizionato $\boldsymbol{\varepsilon}_\theta(\mathbf{x}_t, t, \emptyset)$ (random dropout della condizione durante il training):

$$\tilde{\boldsymbol{\varepsilon}}_\theta(\mathbf{x}_t, t, \mathbf{c}) = \boldsymbol{\varepsilon}_\theta(\mathbf{x}_t, t, \emptyset) + w \cdot [\boldsymbol{\varepsilon}_\theta(\mathbf{x}_t, t, \mathbf{c}) - \boldsymbol{\varepsilon}_\theta(\mathbf{x}_t, t, \emptyset)]$$

Il parametro $w > 1$ è la guidance scale (tipicamente $w = 7.5$ in Stable Diffusion). Incrementare w aumenta la fedeltà alla condizione a spese della diversità — tradeoff qualità/variabilità regolabile a inference time.

Interpretazione: il termine $\boldsymbol{\varepsilon}_\theta(\mathbf{x}_t, t, \mathbf{c}) - \boldsymbol{\varepsilon}_\theta(\mathbf{x}_t, t, \emptyset)$ è la "direzione" nel spazio del rumore che porta verso immagini che soddisfano la condizione $\mathbf{c}$. Si amplifica questa direzione di un fattore w — analogia con la discesa del gradiente nella direzione del classifier implicito.

108b.3 Latent Diffusion Models e Stable Diffusion

LDM (Rombach et al., 2022) applica la diffusione non nello spazio dei pixel ma in uno spazio latente compresso. Un encoder VQ-VAE $\mathcal{E} : \mathbf{x} \to \mathbf{z}$ (fattore di compressione 4–8×) comprime l'immagine; il decoder $\mathcal{D} : \mathbf{z} \to \mathbf{x}$ ricostruisce. Il modello di diffusione $\boldsymbol{\varepsilon}_\theta$ opera su $\mathbf{z}$:

$$\mathcal{L}_{\text{LDM}} = \mathbb{E}_{\mathbf{z}=\mathcal{E}(\mathbf{x}),\boldsymbol{\varepsilon},t}\left[\|\boldsymbol{\varepsilon} - \boldsymbol{\varepsilon}_\theta(\mathbf{z}_t, t, \tau_\theta(\mathbf{y}))\|^2\right]$$

dove τ_θ è un encoder del testo (CLIP o T5). Il costo computazionale si riduce di $(f_{\text{compress}})^2\times$ — da $64 \times 64 \times 3$ a $8 \times 8 \times 4$ per immagini 64×64. Stable Diffusion usa $f = 8$ con $\mathbf{z} \in \mathbb{R}^{64\times64\times4}$ per immagini 512×512.

108b.4 Score Matching e Connessione con DDPM

Lo score di una distribuzione $p(\mathbf{x})$ è $\nabla_{\mathbf{x}} \log p(\mathbf{x})$. Il **denoising score matching** (Vincent, 2011) stima lo score senza calcolare la normalizzante:

$$\mathcal{L}_{\text{DSM}} = \mathbb{E}_{\mathbf{x},\sigma,\tilde{\mathbf{x}}}\left[\|\mathbf{s}_\theta(\tilde{\mathbf{x}}, \sigma) - \nabla_{\tilde{\mathbf{x}}} \log p_\sigma(\tilde{\mathbf{x}}|\mathbf{x})\|^2\right]$$

Con rumore gaussiano $p_\sigma(\tilde{\mathbf{x}}|\mathbf{x}) = \mathcal{N}(\tilde{\mathbf{x}}; \mathbf{x}, \sigma^2\mathbf{I})$: $\nabla_{\tilde{\mathbf{x}}} \log p_\sigma(\tilde{\mathbf{x}}|\mathbf{x}) = -(\tilde{\mathbf{x}} - \mathbf{x})/\sigma^2 = -\boldsymbol{\varepsilon}/\sigma$. Quindi $\mathbf{s}_\theta(\tilde{\mathbf{x}}, \sigma) \approx -\boldsymbol{\varepsilon}/\sigma$ — la rete di score è proporzionale alla noise-predicting network di DDPM. Questa connessione unifica DDPM, SGM (Score-based Generative Models) e il framework SDE del Vol. II (§76).

MDP, Equazioni di Bellman e Algoritmi di Programmazione Dinamica

109.1 Markov Decision Process: Definizione Formale

Il **Markov Decision Process** (MDP) è il linguaggio matematico universale del Reinforcement Learning — il framework che formalizza qualsiasi problema in cui un agente interagisce sequenzialmente con un ambiente cercando di massimizzare una ricompensa cumulativa. Un MDP è specificato dalla quintupla $(\mathcal{S}, \mathcal{A}, P, R, \gamma)$, ciascun elemento della quale cattura un aspetto essenziale dell'interazione agente-ambiente:

- $\mathcal{S}$ è lo spazio degli stati
- $\mathcal{A}$ è lo spazio delle azioni
- $P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ è la funzione di transizione: $P(s'|s, a) = \mathbb{P}[S_{t+1} = s' | S_t = s, A_t = a]$
- $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ è la funzione di reward: $r_t = R(s_t, a_t)$
- $\gamma \in [0, 1)$ è il fattore di sconto

La **proprietà di Markov** — che dà il nome al processo — è l'assunzione strutturale fondamentale: $P(s_{t+1}|s_t, a_t, s_{t-1}, a_{t-1}, \ldots) = P(s_{t+1}|s_t, a_t)$. In altre parole, il futuro è condizionalmente indipendente dall'intera storia dato solo lo stato corrente e l'azione corrente. Lo stato s_t deve quindi riassumere tutta l'informazione rilevante del passato — una condizione che nella pratica è spesso un'idealizzazione, ma che rende il problema analiticamente trattabile.

Una **policy** $\pi : \mathcal{S} \to \Delta(\mathcal{A})$ è la strategia dell'agente: mappa ogni stato in una distribuzione di probabilità sulle azioni disponibili. L'obiettivo del RL è trovare la policy ottimale π^* che massimizza il **return** atteso — il rendimento cumulativo scontato a partire dal tempo t, che assegna peso decrescente ai reward futuri tramite il fattore di sconto $\gamma \in [0, 1)$:

$$G_t = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots$$

109.2 Equazioni di Bellman

La **state-value function** $V^\pi(s) = \mathbb{E}_\pi[G_t|S_t = s]$ misura quanto è vantaggioso, in media, trovarsi nello stato s seguendo la policy π. La sua proprietà fondamentale è soddisfare la **Bellman expectation equation**, una relazione di ricorrenza che lega il valore di uno stato a quello degli stati successori — la chiave per tutti gli algoritmi di programmazione dinamica nel RL:

$$V^\pi(s) = \sum_a \pi(a|s) \sum_{s'} P(s'|s,a)[R(s,a) + \gamma V^\pi(s')]$$

La **action-value function** (o Q-function) $Q^\pi(s,a) = \mathbb{E}_\pi[G_t|S_t = s, A_t = a]$ condiziona sia sullo stato che sull'azione intrapresa nel primo passo — una distinzione cruciale che permette di valutare e confrontare le azioni disponibili senza conoscere esplicitamente il modello di transizione. Anche essa soddisfa un'equazione di Bellman:

$$Q^\pi(s,a) = R(s,a) + \gamma \sum_{s'} P(s'|s,a) \sum_{a'} \pi(a'|s') Q^\pi(s',a')$$

L'**equazione di ottimalità di Bellman** caratterizza la value function ottimale $V^*(s) = \max_\pi V^\pi(s)$ — il massimo valore raggiungibile da qualsiasi policy. A differenza dell'equazione di aspettazione, questa è non-lineare (a causa del max) e non ammette soluzione diretta per sostituzione, ma è il punto di partenza di tutti gli algoritmi di ottimizzazione nel RL:

$$V^*(s) = \max_a \sum_{s'} P(s'|s,a)[R(s,a) + \gamma V^*(s')]$$

$$Q^*(s,a) = R(s,a) + \gamma \sum_{s'} P(s'|s,a) \max_{a'} Q^*(s',a')$$

La policy ottimale è quella greedy rispetto a Q^*: $\pi^*(a|s) = \mathbf{1}[a = \arg\max_{a'} Q^*(s,a')]$. Vale a dire, in ogni stato si sceglie semplicemente l'azione con il massimo Q-value ottimale — nessuna esplorazione è necessaria una volta che Q^* è nota. Il problema del RL si riduce quindi a stimare Q^* in modo accurato.

109.3 Value Iteration e Policy Iteration

Value Iteration è l'algoritmo fondamentale per risolvere un MDP con modello noto. Applica iterativamente l'operatore di Bellman ottimale $\mathcal{T}^*$, partendo da una stima arbitraria V_0 e raffinandola ad ogni passo. La convergenza è garantita dal teorema del punto fisso di Banach:

$$V_{k+1}(s) = \max_a \sum_{s'} P(s'|s,a)[R(s,a) + \gamma V_k(s')]$$

Teorema di Convergenza — Value Iteration

L'operatore $\mathcal{T}^*$ è una contrazione in norma ℓ^∞ con fattore γ: $\|\mathcal{T}^*V - \mathcal{T}^*V'\|_\infty \leq \gamma\|V - V'\|_\infty$. Quindi $V_k \to V^*$ con tasso geometrico, e l'errore dopo k iterazioni è $\|V_k - V^*\|_\infty \leq \frac{\gamma^k}{1-\gamma}\|V_1 - V_0\|_\infty$.

Policy Iteration è un approccio alternativo che spesso converge molto più rapidamente di Value Iteration in termini di numero di iterazioni, pur avendo un costo maggiore per iterazione (richiede la soluzione di un sistema lineare). Alterna due fasi fino a stabilizzazione della policy:

Algoritmo: Policy Iteration

inizializza π_0 arbitrariamente

ripeti:

// Policy Evaluation: risolvi sistema lineare

$V^{\pi_k} \leftarrow (I - \gamma P^{\pi_k})^{-1} R^{\pi_k}$

// Policy Improvement: policy greedy

$\pi_{k+1}(s) \leftarrow \arg\max_a \sum_{s'} P(s'|s,a)[R(s,a) + \gamma V^{\pi_k}(s')]$

finché $\pi_{k+1} = \pi_k$

Policy Iteration converge in un numero finito di passi — al più $|\mathcal{A}|^{|\mathcal{S}|}$ policy deterministiche

distinte — poiché ogni passo di policy improvement garantisce strettamente $V^{\pi_{k+1}}(s) \geq V^{\pi_k}(s)$ per tutti gli s, con uguaglianza solo quando si è raggiunta la policy ottimale. In pratica, Policy Iteration converge tipicamente in poche decine di iterazioni anche per MDP di grandi dimensioni, rendendola spesso preferibile a Value Iteration.

Q-Learning: TD Learning e Convergenza

110.1 Temporal Difference Learning

Quando il modello di transizione P è sconosciuto — la situazione tipica nel RL applicato — non è possibile applicare direttamente le equazioni di Bellman. Si ricorre ai metodi *model-free*, che stimano la value function direttamente dall'esperienza senza costruire un modello esplicito dell'ambiente. TD(0) è l'algoritmo base: aggiorna incrementalmente $V(S_t)$ dopo ogni transizione, utilizzando il **TD error** δ_t come segnale di apprendimento:

$$\delta_t = R_{t+1} + \gamma V(S_{t+1}) - V(S_t) \quad \text{(errore di previsione)}$$

$$V(S_t) \leftarrow V(S_t) + \alpha\, \delta_t$$

Il TD error misura la discrepanza tra la stima corrente $V(S_t)$ e una stima "migliorata" ottenuta osservando una singola transizione — tecnica nota come **bootstrapping**, poiché si usa la stima corrente per aggiornare sé stessa. Questo approccio è notevolmente più efficiente di Monte Carlo, che deve attendere la fine di un intero episodio prima di aggiornare: TD(0) impara online dopo ogni singola transizione. A convergenza, il TD target $R_{t+1} + \gamma V(S_{t+1})$ deve essere uguale in media a $V(S_t)$, condizione che coincide esattamente con la Bellman expectation equation.

110.2 Q-Learning: Aggiornamento e Convergenza

Q-learning (Watkins, 1989) è un algoritmo *off-policy* che stima direttamente la Q-function ottimale Q^*, senza riferimento a nessuna policy specifica. La sua semplicità e l'eleganza teorica della convergenza garantita lo rendono uno degli algoritmi fondamentali del RL:

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha\Big[R_{t+1} + \gamma \max_{a'} Q(S_{t+1}, a') - Q(S_t, A_t)\Big]$$

L'algoritmo è *off-policy* perché il target usa $\max_{a'} Q(S_{t+1}, a')$ — cioè l'azione greedy ottimale — indipendentemente da quale azione viene effettivamente eseguita nel passo successivo (che potrebbe essere ε-greedy o qualsiasi altra policy di esplorazione). Questa separazione tra la policy di comportamento (usata per raccogliere dati) e la policy target (quella che si sta ottimizzando) è ciò che rende Q-learning off-policy, a differenza di SARSA che è on-policy. Il target bootstrap $R_{t+1} + \gamma \max_{a'} Q(S_{t+1}, a')$ approssima $Q^*(S_t, A_t)$ tramite la Bellman optimality equation.

Teorema di Convergenza Q-Learning (Watkins e Dayan, 1992)

La convergenza è garantita quasi certamente sotto tre condizioni: ogni coppia (s, a) deve essere visitata infinitamente spesso (esplorazione sufficiente), la sequenza di learning rate deve soddisfare le condizioni di Robbins-Monro $\sum_t \alpha_t = \infty$ e $\sum_t \alpha_t^2 < \infty$, e i reward devono essere limitati. Queste condizioni assicurano $Q_t(s, a) \to Q^*(s, a)$ per ogni (s, a) — un risultato teorico di straordinaria importanza che, tuttavia, richiede spazio di stati finito e tabellare.

Il dilemma esplorazione-sfruttamento è centrale in ogni algoritmo RL: per stimare Q^* in modo accurato, ogni coppia stato-azione deve essere visitata, ma nella pratica si vuole anche massimizzare la ricompensa accumulata. La strategia ε-greedy bilancia questi obiettivi: con probabilità ε si sceglie un'azione uniformemente casuale (esplorazione), altrimenti si segue $\arg\max_a Q(s, a)$ (sfruttamento). Decrementando ε nel tempo — tipicamente come $1/\sqrt{t}$ o con un decay esponenziale — si sposta progressivamente l'enfasi dallo sfruttamento man mano che la stima Q matura.

Deep Q-Network: Experience Replay e Target Network

111.1 Il Problema: Correlazioni Temporali

Il passaggio da Q-learning tabulare a Q-learning con approssimazione funzionale (rete neurale Q_θ) introduce due problemi di instabilità che si manifestano in modo drammatico nella pratica. Il primo è la **correlazione temporale**: le transizioni consecutive $(S_t, A_t, R_{t+1}, S_{t+1})$ sono fortemente correlate nel tempo — l'agente esplora regioni connesse dello spazio degli stati — violando l'assunzione i.i.d. di SGD e portando il training a oscillare o divergere. Il secondo è il **target non-stazionario**: poiché il target $R_{t+1} + \gamma \max_{a'} Q_\theta(S_{t+1}, a')$ dipende dagli stessi parametri θ che si aggiornano, ogni modifica alla rete sposta simultaneamente tutti i target, creando un inseguimento instabile.

DQN (Mnih et al., 2013-2015) risolve entrambi questi problemi con due innovazioni architetturali che sono diventate componenti standard del deep RL: l'*experience replay* e la *target*

network.

111.2 Experience Replay e Target Network

Experience Replay: si memorizza ogni transizione (s, a, r, s') in un replay buffer $\mathcal{D}$ di capacità N. Ad ogni step si campiona un mini-batch casuale $\{(s_i, a_i, r_i, s'_i)\}_{i=1}^{B}$ — rompe le correlazioni temporali.

Target Network: si mantiene una copia "congelata" Q_{θ^-} dei parametri della rete, aggiornata ogni C step copiando i pesi dalla rete principale Q_θ. I target del training vengono calcolati usando Q_{θ^-}, non Q_θ:

$$y_i = r_i + \gamma \max_{a'} Q_{\theta^-}(s'_i, a')$$

$$\mathcal{L}(\theta) = \frac{1}{B} \sum_i (y_i - Q_\theta(s_i, a_i))^2$$

La target network trasforma il bersaglio mobile in un bersaglio quasi-stazionario: i target rimangono fissi per C step (tipicamente $C = 10000$), poi si aggiornano in un singolo salto. Questo rompe il ciclo di feedback problematico in cui la rete che si aggiorna modifica immediatamente i propri target, causando oscillazioni.

Double DQN: DQN soffre di overestimation perché $\max_{a'} Q_{\theta^-}$ è polarizzato verso il massimo rumoroso. Double DQN separa selezione e valutazione:

$$y_i = r_i + \gamma Q_{\theta^-}\left(s'_i, \arg\max_{a'} Q_\theta(s'_i, a')\right)$$

Policy Gradient e REINFORCE

112.1 Policy Gradient Theorem

I metodi *policy gradient* adottano un approccio radicalmente diverso rispetto ai metodi value-based come Q-learning: invece di stimare Q^* e poi derivarne una policy greedy, ottimizzano direttamente i parametri θ della policy π_θ. Questo approccio ha il vantaggio di funzionare naturalmente con azioni continue e di convergere a policy stocastiche ottimali, che in certi giochi

parzialmente osservabili sono superiori a quelle deterministiche. L'obiettivo da massimizzare è il valore atteso del return:

$$J(\theta) = \mathbb{E}_{\pi_\theta}[G_0] = \sum_s d^{\pi_\theta}(s) \sum_a \pi_\theta(a|s) Q^{\pi_\theta}(s,a)$$

dove $d^{\pi_\theta}(s)$ è la distribuzione stazionaria degli stati sotto π_θ — la frequenza con cui l'agente visita ogni stato seguendo la policy. Questo obiettivo non è direttamente differenziabile in modo semplice, poiché la distribuzione degli stati d^{π_θ} dipende anch'essa da θ. Il Policy Gradient Theorem risolve elegantemente questo problema.

Policy Gradient Theorem (Sutton et al., 1999)

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}[\nabla_\theta \log \pi_\theta(A_t|S_t) \cdot Q^{\pi_\theta}(S_t, A_t)]$$

La derivazione del gradiente di J rispetto a θ è il cuore del framework. Si usa l'identità del log-derivative — che esprime il gradiente di una distribuzione parametrizzata attraverso il gradiente del suo logaritmo: $\nabla_\theta \pi_\theta(a|s) = \pi_\theta(a|s) \nabla_\theta \log \pi_\theta(a|s)$. Differenziando J e applicando questa identità:

$\nabla_\theta J = \sum_s d^{\pi_\theta}(s) \sum_a \nabla_\theta [\pi_\theta(a|s) Q^{\pi_\theta}(s,a)]$ Linearità del gradiente (trascurando la dipendenza di d^{π_θ} da θ — dimostrabile che il contributo è zero)

$= \sum_s d^{\pi_\theta}(s) \sum_a \pi_\theta(a|s) \nabla_\theta \log \pi_\theta(a|s) \cdot Q^{\pi_\theta}(s,a)$ Log-derivative trick: $\nabla \pi = \pi \nabla \log \pi$

$= \mathbb{E}_{\pi_\theta}[\nabla_\theta \log \pi_\theta(A|S) \cdot Q^{\pi_\theta}(S,A)]$ Definizione di valore atteso rispetto a $(S \sim d^{\pi_\theta}, A \sim \pi_\theta(\cdot|S))$

112.2 REINFORCE e Riduzione della Varianza

REINFORCE (Williams, 1992) è l'algoritmo più diretto per implementare il Policy Gradient Theorem: stima il gradiente ∇J con Monte Carlo, campionando intere traiettorie e usandone il return come stima del Q-value. Dato un episodio completo $(S_0, A_0, R_1, S_1, \ldots, S_{T-1}, A_{T-1}, R_T)$, l'aggiornamento è:

$$\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{n=1}^{N} \sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(A_t^{(n)}|S_t^{(n)}) \cdot G_t^{(n)}$$

Lo stimatore REINFORCE è unbiased — in aspettazione converge al vero gradiente — ma soffre di alta varianza: G_t fluttua enormemente tra episodi diversi, poiché dipende da tutta la catena di reward futuri in un ambiente stocastico. Questa varianza elevata rallenta fortemente la convergenza. L'introduzione di una **baseline** $b(s)$ — una funzione arbitraria dello stato, tipicamente $V^{\pi_\theta}(s)$ — riduce la varianza senza introdurre bias nel gradiente:

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}[\nabla_\theta \log \pi_\theta(A|S) \cdot (Q^{\pi_\theta}(S,A) - b(S))]$$

La baseline non introduce bias perché, per costruzione, $\mathbb{E}_{\pi_\theta}[\nabla \log \pi_\theta(A|S) \cdot b(S)] = b(S) \sum_a \nabla_\theta \pi_\theta(a|S) = b(S) \nabla_\theta \sum_a \pi_\theta(a|S) = b(S) \nabla_\theta 1 = 0$. La scelta ottimale (minimo varianza) è $b^*(s) = \frac{\mathbb{E}[\|\nabla \log \pi\|^2 Q]}{\mathbb{E}[\|\nabla \log \pi\|^2]}$ — in pratica si usa $V^{\pi_\theta}(s)$.

Actor-Critic e Generalized Advantage Estimation

113.1 Actor-Critic: Architettura

L'architettura Actor-Critic rappresenta la sintesi naturale tra policy gradient e value-based methods: l'**actor** (π_θ) apprende la policy, mentre il **critic** (V_w) stima il valore degli stati, fornendo un segnale a varianza ridotta per guidare l'aggiornamento dell'actor. Al centro del framework vi è l'**advantage function**, che misura quanto un'azione è migliore o peggiore della media in quello stato:

$$A^\pi(s,a) = Q^\pi(s,a) - V^\pi(s)$$

Sostituire il Q-value con l'advantage nella policy gradient riduce la varianza del gradiente — poiché A è centrata attorno allo zero — mantenendo il gradiente unbiased. L'aggiornamento dell'actor diventa quindi:

$$\nabla_\theta J \approx \nabla_\theta \log \pi_\theta(a|s) \cdot A^\pi(s,a) \approx \nabla_\theta \log \pi_\theta(a|s) \cdot \delta_t$$

dove $\delta_t = R_{t+1} + \gamma V_w(S_{t+1}) - V_w(S_t)$ è il TD error — un estimatore one-step dell'advantage, computazionalmente efficiente perché non richiede di attendere la fine dell'episodio. Il critic V_w viene aggiornato simultaneamente minimizzando l'errore quadratico rispetto al TD target:

$\|V_w(S_t) - (R_{t+1} + \gamma V_w(S_{t+1}))\|^2$. I due componenti si aggiornano in modo coordinato ad ogni passo, con il critic che guida l'actor verso azioni migliori della media.

A2C (Synchronous Advantage Actor-Critic) estende Actor-Critic alla raccolta parallela di esperienze: N ambienti distinti eseguono in parallelo la policy corrente e raccolgono batch di transizioni sincronamente. Actor e critic vengono poi aggiornati con il batch combinato di tutti gli ambienti. La raccolta parallela riduce la correlazione tra le esperienze (simile all'experience replay ma senza memoria esplicita) e aumenta la dimensione effettiva del batch, stabilizzando il training.

113.2 Generalized Advantage Estimation (GAE)

Il **Generalized Advantage Estimation** (GAE, Schulman et al., 2016) risolve il dilemma fondamentale della stima dell'advantage: i ritorni Monte Carlo a lungo orizzonte hanno bassa varianza ma alto bias (perché il critic è imperfetto); gli stimatori one-step hanno alta varianza ma basso bias. GAE introduce un parametro $\lambda \in [0, 1]$ che interpola tra questi estremi. Si definiscono i k-step TD error come:

$$\hat{A}_t^{(k)} = -V(S_t) + R_{t+1} + \gamma R_{t+2} + \cdots + \gamma^{k-1} R_{t+k} + \gamma^k V(S_{t+k})$$

GAE combina questi stimatori in una media esponenzialmente pesata, assegnando pesi decrescenti agli stimatori a lungo orizzonte:

$$\hat{A}_t^{\text{GAE}} = (1-\lambda) \sum_{k=0}^{\infty} \lambda^k \hat{A}_t^{(k+1)} = \sum_{l=0}^{\infty} (\gamma\lambda)^l \delta_{t+l}$$

dove $\delta_t = R_{t+1} + \gamma V(S_{t+1}) - V(S_t)$ è il TD error. I casi estremi rivelano l'interpretazione intuitiva: per $\lambda = 0$, $\hat{A}_t = \delta_t$ (alta bias, bassa varianza — dipende da un solo passo); per $\lambda = 1$, $\hat{A}_t = \sum_l \gamma^l \delta_{t+l} = G_t - V(S_t)$ (Monte Carlo, basso bias, alta varianza). Con $\lambda \in (0, 1)$ si ottiene il compromesso desiderato: tipicamente $\lambda = 0.95$ offre un bilanciamento robusto tra bias e varianza in una grande varietà di ambienti.

PPO: Proximal Policy Optimization — Derivazione Completa

114.1 Dal TRPO al Clipped Objective

TRPO (Trust Region Policy Optimization) è il predecessore di PPO che ha formalmente dimostrato perché i grandi aggiornamenti di policy portano a instabilità: massimizzare il surrogate objective senza limitare la distanza dalla policy precedente può peggiorare irrecuperabilmente le performance. TRPO risolve questo con un vincolo esplicito sulla divergenza KL:

$$\max_{\theta} \ \mathbb{E}_t\left[\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}\hat{A}_t\right] \quad \text{s.t. } \mathbb{E}_t[D_{\text{KL}}(\pi_{\theta_{\text{old}}}\|\pi_\theta)] \leq \delta$$

Il vincolo KL limita quanto la policy può cambiare in un singolo aggiornamento, impedendo degrado catastrofico delle performance. Tuttavia, TRPO ha un costo computazionale proibitivo per reti moderne: richiede l'inversa della Fisher Information Matrix — equivalente alla Hessiana della KL — la cui computazione ha costo cubico nel numero di parametri. Con milioni di parametri, questo è semplicemente impraticabile.

PPO (Proximal Policy Optimization, Schulman et al., 2017) raggiunge la stessa stabilità di TRPO con una frazionde del costo computazionale. L'idea è sostituire il vincolo KL esplicito con un semplice *clipping* sull'importance ratio $r_t(\theta) = \pi_\theta(a_t|s_t)/\pi_{\theta_{\text{old}}}(a_t|s_t)$, che misura quanto la nuova policy si discosta da quella vecchia:

$$\mathcal{L}^{\text{CLIP}}(\theta) = \mathbb{E}_t\Big[\min\Big(r_t(\theta)\hat{A}_t,\ \text{clip}(r_t(\theta), 1-\varepsilon, 1+\varepsilon)\hat{A}_t\Big)\Big]$$

Analisi del clipping

L'effetto del clipping è asimmetrico e dipende crucialmente dal segno dell'advantage $\hat{A}_t$, il che garantisce aggiornamenti conservativi in entrambe le direzioni:

Caso	$r_t(\theta)$ **favorito da** $\mathcal{L}$	**Clip attivo quando**	**Effetto**
$\hat{A}_t > 0$ (buona azione)	Aumentare r_t (azione più probabile)	$r_t > 1+\varepsilon$	Limita l'aumen
$\hat{A}_t < 0$ (cattiva azione)	Diminuire r_t (azione meno probabile)	$r_t < 1-\varepsilon$	Limita la dimin

Il min garantisce che l'obiettivo clippato sia sempre un *lower bound pessimistico* del surrogate non-clippato: la policy può migliorare l'obiettivo solo entro i limiti della trust region implicita definita da ε (tipicamente 0.1–0.2). Questo rende PPO notevolmente stabile anche senza vincoli

espliciti sulla divergenza KL, e spiega perché è diventato l'algoritmo preferito per RLHF nei grandi modelli linguistici.

Obiettivo completo PPO

$$\mathcal{L}^{\mathrm{PPO}} = \mathbb{E}_t\big[\mathcal{L}_t^{\mathrm{CLIP}} - c_1\mathcal{L}_t^{\mathrm{VF}} + c_2 H[\pi_\theta](s_t)\big]$$

dove $\mathcal{L}_t^{\mathrm{VF}} = (V_\theta(s_t) - V_t^{\mathrm{targ}})^2$ è la loss del critic e $H[\pi_\theta]$ è il bonus di entropia che penalizza le policy deterministiche, mantenendo una pressione costante verso l'esplorazione durante il training. I pesi tipici $c_1 = 0.5$ e $c_2 = 0.01$ funzionano bene su una vasta gamma di problemi senza richiedere ottimizzazione specifica per ogni task.

SAC: Soft Actor-Critic e Maximum Entropy RL

115.1 Framework Maximum Entropy

Il framework del **Maximum Entropy RL** introduce una prospettiva diversa sull'obiettivo di ottimizzazione: invece di massimizzare solo il reward atteso, si aggiunge un termine di entropia che penalizza le policy troppo deterministiche. Questa formulazione incoraggia l'esplorazione, produce policy più robuste alle perturbazioni del modello, e in certi ambienti porta a soluzioni qualitativamente migliori rispetto al RL standard:

$$J(\pi) = \sum_{t=0}^{T} \mathbb{E}_{(s_t,a_t)\sim\rho_\pi}[R(s_t,a_t) + \alpha H(\pi(\cdot|s_t))]$$

dove $\alpha > 0$ è il parametro di temperatura e $H(\pi(\cdot|s)) = -\mathbb{E}_{a\sim\pi}[\log \pi(a|s)]$ è l'entropia della policy in s. La policy ottimale bilancia alta ricompensa e alta entropia: rimane stocastica anche a convergenza, preferendo azioni diverse a meno che una non sia chiaramente superiore alle altre — un comportamento qualitativamente diverso da quello del RL standard che converge sempre a policy deterministiche.

Soft Bellman Equations

Le equazioni di Bellman si modificano in modo naturale per incorporare il termine entropico. La **soft value function** aggiunge l'entropia attesa al valore standard, creando un obiettivo composito che bilancia reward futuro e diversità delle azioni:

$$V^\pi(s) = \mathbb{E}_{a\sim\pi}[Q^\pi(s,a) - \alpha \log \pi(a|s)]$$

$$Q^\pi(s,a) = R(s,a) + \gamma \mathbb{E}_{s'\sim P}[V^\pi(s')]$$

La policy ottimale soft è esplicita: $\pi^*(a|s) \propto \exp(Q^*(s,a)/\alpha)$ — una softmax del Q-value. SAC mantiene due Q-network $Q_{\theta_1}, Q_{\theta_2}$ (per ridurre l'overestimation) e aggiorna:

$$\mathcal{L}_Q = \mathbb{E}_{(s,a,r,s')\sim\mathcal{D}}\left[\left(Q_\theta(s,a) - (r + \gamma \hat{V}(s'))\right)^2\right]$$

$$\hat{V}(s') = \min_{i=1,2} Q_{\theta_i^-}(s', \hat{a}') - \alpha \log \pi_\phi(\hat{a}'|s'), \quad \hat{a}' \sim \pi_\phi(\cdot|s')$$

La temperature α può essere appresa automaticamente minimizzando $\mathbb{E}_{a\sim\pi}[-\alpha \log \pi(a|s) - \alpha\mathcal{H}_{\text{target}}]$ dove $\mathcal{H}_{\text{target}} = -|\mathcal{A}|$ è una target entropy.

Model-Based RL: Dyna e World Models

116.1 Dyna-Q: Pianificazione con Modello Imparato

Dyna (Sutton, 1991) è il framework seminale per l'integrazione di RL model-free con pianificazione tramite modello imparato. L'intuizione fondamentale è che le esperienze reali con l'ambiente sono costose (tempo, risorse, rischio), ma una volta appreso un modello $\hat{P}(s'|s,a)$ e $\hat{R}(s,a)$, si possono generare esperienze simulate gratuitamente per migliorare la policy. Ad ogni step reale, Dyna esegue tre operazioni in sequenza:

Algoritmo: Algoritmo Dyna-Q

osserva (s, a, r, s') dall'ambiente reale

Aggiorna $Q(s,a)$ con Q-learning reale

Aggiorna il modello $\hat{P}, \hat{R}$ con (s, a, r, s')

ripeti n volte:

Campiona $(s_{\text{sim}}, a_{\text{sim}})$ casualmente dalla memoria

$r_{\text{sim}}, s'_{\text{sim}} \leftarrow \hat{R}(s_{\text{sim}}, a_{\text{sim}}),\ \hat{P}(\cdot|s_{\text{sim}}, a_{\text{sim}})$

Aggiorna $Q(s_{\text{sim}}, a_{\text{sim}})$ con step simulato

I n step simulati per ogni step reale moltiplicano l'efficienza campionaria di un fattore $n+1$: si estrae molto più apprendimento da ogni costosa interazione reale. L'accuratezza del modello è cruciale — errori sistematici nel modello si propagano e amplificano attraverso la pianificazione, un fenomeno noto come "model bias" che limita il numero di step simulati utilizzabili.

World Models (Ha e Schmidhuber, 2018)

I World Models hanno tre componenti: un encoder visivo E (VAE), un modello di transizione M (MDN-RNN, mixture density network + LSTM), e un controller C (piccola rete). Il controller opera nello spazio latente $\mathbf{z}$ invece dello spazio delle osservazioni — riduce la dimensionalità e permette pianificazione veloce:

$$\mathbf{z}_t = E(\mathbf{o}_t), \quad (\mathbf{z}_{t+1}, \hat{r}_t) = M(\mathbf{z}_t, \mathbf{a}_t, \mathbf{h}_t), \quad \mathbf{a}_t = C(\mathbf{z}_t, \mathbf{h}_t)$$

La MDN modella $p(\mathbf{z}_{t+1}|\mathbf{z}_t, \mathbf{a}_t, \mathbf{h}_t)$ come miscela gaussiana — cattura la multimodalità del futuro. Il controller può essere addestrato interamente nei "sogni" (simulazioni del world model) senza interagire con l'ambiente reale.

RL Avanzato: Multi-Agent, Offline RL e RLHF

116b.1 Rainbow DQN: Combinazione di 6 Miglioramenti

Rainbow (Hessel et al., 2018) combina sei estensioni di DQN in un unico agente, mostrando che i miglioramenti sono complementari:

Componente	Innovazione chiave	Miglioramento
Double DQN	Separa selezione e valutazione	Riduce overestimation
Prioritized Experience Replay	Campiona transizioni con $\lvert_t\rvert$ alto	Sample efficiency
Dueling Networks	$Q(s,a) = V(s) + A(s,a) - \bar{A}$	Stima V più stabile
Multi-step Returns	n-step TD target	Bias-variance tradeoff
Distributional RL (C51)	Distribuzione del return $Z(s,a)$ invece di Q	Gestisce multimodalità
Noisy Networks	$\mathbf{w} = \boldsymbol{\mu} + \boldsymbol{\sigma} \odot \boldsymbol{\varepsilon}$ per esplorazione	Sostituisce ε-greedy

Distributional RL: C51

C51 (Bellemare et al., 2017) modella la distribuzione completa del return $Z^\pi(s,a)$ invece del valore atteso $Q^\pi(s,a)$. La distribuzione è discretizzata su $C = 51$ atomi $\{z_i\}_{i=1}^{51}$ con $z_i \in [-V_{\max}, V_{\max}]$:

$$Z^\pi(s,a) = \sum_{i=1}^{51} p_i(s,a) \cdot \delta_{z_i}$$

L'aggiornamento proietta la distribuzione del Bellman target $\mathcal{T}Z = R + \gamma Z(S', A')$ sulla griglia fissa tramite distribuzione dell'operatore di Bellman:

$$(\Phi \hat{\mathcal{T}} Z_\theta)_i = \sum_j \left[1 - \frac{|\text{clip}(\hat{\mathcal{T}} z_j, z_1, z_{51}) - z_i|}{\Delta z} \right]^+ p_j(s', a^*)$$

La loss è la cross-entropy tra la distribuzione proiettata e la predizione del modello — minimizza la distanza di Wasserstein in forma chiusa per distribuzioni discrete.

116b.2 Prioritized Experience Replay

PER (Schaul et al., 2016) campiona transizioni proporzionalmente al loro TD error assoluto:

$$P(i) = \frac{p_i^\alpha}{\sum_k p_k^\alpha}, \qquad p_i = |\delta_i| + \epsilon$$

dove $\alpha \in [0,1]$ controlla il grado di prioritizzazione ($\alpha = 0$: uniform sampling, $\alpha = 1$: full prioritization). Per correggere il bias introdotto dal sampling non-uniforme, si usano importance

sampling weights:

$$w_i = \left(\frac{1}{N \cdot P(i)}\right)^{\beta}, \qquad \beta : 0.4 \to 1 \text{ (annealing)}$$

L'aggiornamento diventa $\theta \leftarrow \theta + \alpha w_i \delta_i \nabla_\theta Q_\theta$. L'implementazione efficiente usa un **sum tree** (binary tree con la somma delle priorità nei nodi interni) per campionare e aggiornare in $O(\log N)$.

116b.3 Offline RL: Conservative Q-Learning (CQL)

Nell'offline RL, il dataset di interazioni $\mathcal{D}$ è fisso (nessuna interazione con l'ambiente). Il problema principale: Q-learning off-policy tende a overestimare $Q^*(s,a)$ per azioni a non coperte dal dataset — porta a policy che selezionano azioni distribuzionalmente fuori da $\mathcal{D}$ con Q-values artificialmente alti.

CQL (Kumar et al., 2020) augmenta la standard Bellman backup con un termine di penalità che minimizza il valore atteso sotto la policy corrente e massimizza il valore sui dati:

$$\min_Q \underbrace{\alpha(\mathbb{E}_{s\sim\mathcal{D},a\sim\pi_\theta}[Q(s,a)] - \mathbb{E}_{s,a\sim\mathcal{D}}[Q(s,a)])}_{\text{penalità CQL}} + \underbrace{\frac{1}{2}\mathbb{E}_{s,a,s'\sim\mathcal{D}}\left[(Q(s,a) - (r + \gamma\hat{V}(s')))^2\right]}_{\text{Bellman error}}$$

Teorema CQL (Kumar et al., 2020)

L'obiettivo CQL produce una lower bound sulla vera Q^π: $\hat{Q}^\pi(s,a) \leq Q^\pi(s,a)$ per tutti (s,a) nel supporto di $\mathcal{D}$. Di conseguenza, la policy $\hat{\pi} = \arg\max_a \hat{Q}^\pi(s,a)$ è sicura — non seleziona azioni con Q-values overestimati.

116b.4 RLHF: Reinforcement Learning from Human Feedback

RLHF (Christiano et al., 2017; Ouyang et al., 2022 — InstructGPT) allinea LLM alle preferenze umane in tre fasi:

Fase 1 — Supervised Fine-Tuning (SFT): il modello base viene fine-tuned su prompt-risposta dimostrati da umani: $\mathcal{L}_{\text{SFT}} = -\sum_t \log \pi_\theta(y_t | x, y_{<t})$.

Fase 2 — Reward Model Training: per coppie di risposte (y_1, y_2) confrontate da un annotatore umano con preferenza $y_1 \succ y_2$, si addestra un reward model $r_\phi(x,y)$ con la Bradley-Terry

loss:

$$\mathcal{L}_{\text{RM}} = -\mathbb{E}_{(x,y_1,y_2)}[\log \sigma(r_\phi(x,y_1) - r_\phi(x,y_2))]$$

Fase 3 — RL con PPO: si ottimizza la policy π_θ per massimizzare il reward del RM con penalità KL per non allontanarsi troppo da π_{SFT}:

$$\max_\theta \mathbb{E}_{x\sim\mathcal{D},y\sim\pi_\theta(y|x)}[r_\phi(x,y) - \beta D_{\text{KL}}(\pi_\theta(\cdot|x)\|\pi_{\text{SFT}}(\cdot|x))]$$

La penalità KL con peso β previene il reward hacking — il modello non trova modi di massimizzare r_ϕ che non corrispondono alla preferenza umana reale (es. risposte lunghe e ripetitive che ottengono alto reward dal RM ma sono di scarsa qualità).

116b.5 Multi-Armed Bandit e Exploration-Exploitation

Il Multi-Armed Bandit (MAB) è il caso degenere dell'MDP con un solo stato e K azioni (braccia). Ogni braccio k restituisce reward $r \sim \nu_k$ con media μ_k sconosciuta. L'obiettivo è massimizzare il reward cumulativo — equivale a minimizzare il **regret**:

$$R_T = T\mu^* - \sum_{t=1}^{T} \mathbb{E}[r_t], \quad \mu^* = \max_k \mu_k$$

Il Lower bound di Lai e Robbins (1985): qualsiasi strategia ragionevole ha $R_T \geq \Omega\Big(\sum_{k:\mu_k<\mu^*} \frac{\log T}{\Delta_k}\Big)$ dove $\Delta_k = \mu^* - \mu_k$. L'algoritmo UCB1 raggiunge questo bound:

$$A_t = \arg\max_k \left[\hat{\mu}_k + \sqrt{\frac{2\log t}{N_k(t)}}\right]$$

dove $\hat{\mu}_k$ è la media empirica e $N_k(t)$ il numero di pulls. Il termine $\sqrt{2\log t/N_k}$ è l'Upper Confidence Bound — incentiva l'esplorazione di braccia poco campionate.

InfoNCE e NT-Xent: Fondamenti delle Loss Contrastive

117.1 InfoNCE: Derivazione dalla Mutual Information

L'obiettivo del contrastive learning è apprendere una funzione f tale che le rappresentazioni di viste diverse dello stesso dato siano simili. InfoNCE (van den Oord et al., 2018) massimizza un lower bound dell'informazione mutua $I(\mathbf{x}; \mathbf{c})$ tra il contesto $\mathbf{c}$ e le osservazioni future $\mathbf{x}$.

Dato un batch di N coppie di viste $\{(\mathbf{x}_i, \mathbf{x}_i^+)\}_{i=1}^N$ (positive pairs dallo stesso dato) e $N-1$ negativi per ogni ancora, la loss InfoNCE:

$$\mathcal{L}_{\text{InfoNCE}} = -\frac{1}{N}\sum_{i=1}^{N} \log \frac{\exp(\text{sim}(\mathbf{z}_i, \mathbf{z}_i^+)/\tau)}{\sum_{j=1}^{N} \exp(\text{sim}(\mathbf{z}_i, \mathbf{z}_j^+)/\tau)}$$

dove $\text{sim}(\mathbf{u}, \mathbf{v}) = \mathbf{u}^T\mathbf{v}/(\|\mathbf{u}\|\|\mathbf{v}\|)$ è la cosine similarity e $\tau > 0$ è la temperatura che controlla la concentrazione della distribuzione. Temperature basse rendono la loss più selettiva (alta precisione ma possibile instabilità); temperature alte ammorbidiscono la distribuzione. Il bound sulla mutual information:

Teorema — InfoNCE come Lower Bound su MI

$$I(\mathbf{x}; \mathbf{c}) \geq \log N - \mathcal{L}_{\text{InfoNCE}}$$

Massimizzare $-\mathcal{L}_{\text{InfoNCE}}$ equivale a massimizzare un lower bound sulla mutual information $I(\mathbf{x}; \mathbf{c})$ tra la rappresentazione e il contesto. Il bound è tanto più stretto quanto maggiore è il numero di negativi N: al crescere di N, la loss fornisce un segnale di apprendimento più informativo, ma il costo computazionale scala linearmente. Questa dipendenza dal batch size è la motivazione principale per le alternative che vedremo (MoCo, BYOL).

NT-Xent: Simmetria Bidirezionale

NT-Xent (Normalized Temperature-scaled Cross Entropy), la loss adottata in SimCLR, è la versione simmetrica bidirezionale di InfoNCE: la loss viene calcolata sia nella direzione $i \to j$ che $j \to i$ e poi mediata, fornendo un segnale di apprendimento più ricco e stabile. Con $2N$ esempi nel batch (ogni immagine ha due augmented views):

$$\ell(i, j) = -\log \frac{\exp(\text{sim}(\mathbf{z}_i, \mathbf{z}_j)/\tau)}{\sum_{k=1}^{2N} \mathbf{1}[k \neq i] \exp(\text{sim}(\mathbf{z}_i, \mathbf{z}_k)/\tau)}$$

$$\mathcal{L}_{\text{NT-Xent}} = \frac{1}{2N}\sum_{k=1}^{N}[\ell(2k-1,2k) + \ell(2k,2k-1)]$$

I $2N-2$ negativi per ogni esempio includono tutte le altre immagini del batch — sia le prime che le seconde augmentazioni — eccetto la coppia positiva corrente. Questo significa che batch molto grandi (SimCLR usa tipicamente $N = 4096$ o $N = 8192$) sono critici per ottenere buone performance: con pochi negativi, il task diventa troppo facile e il modello non impara rappresentazioni discriminative.

SimCLR: Simple Framework for Contrastive Learning

118.1 Pipeline Completa

SimCLR (Chen et al., 2020) dimostra che un framework contrastivo semplice può raggiungere performance paragonabili alla supervisione completa, identificando con precisione i fattori critici del contrastive learning. Il sistema si articola in quattro componenti essenziali che devono essere presenti simultaneamente per ottenere rappresentazioni di qualità:

1. **Augmentation:** per ogni immagine $\mathbf{x}$, si campionano due trasformazioni $t, t' \sim \mathcal{T}$ e si ottengono $\tilde{\mathbf{x}}_i = t(\mathbf{x})$, $\tilde{\mathbf{x}}_j = t'(\mathbf{x})$ — la coppia positiva
2. **Encoder:** $\mathbf{h} = f(\tilde{\mathbf{x}})$ — tipicamente ResNet-50, $\mathbf{h} \in \mathbb{R}^{2048}$
3. **Projection head:** $\mathbf{z} = g(\mathbf{h}) = \mathbf{W}^{(2)}\sigma(\mathbf{W}^{(1)}\mathbf{h})$ — MLP con 1 layer nascosto, $\mathbf{z} \in \mathbb{R}^{128}$
4. **Loss:** NT-Xent su $\mathbf{z}$ (non su $\mathbf{h}$)

Il projection head è forse il risultato empirico più sorprendente di SimCLR: la loss contrastiva viene applicata a $\mathbf{z}$ (output del projection head MLP), mentre per i task a valle si usa $\mathbf{h}$ (output dell'encoder). Rimuovere il projection head causa un calo di 10 punti percentuali sulle downstream task. La spiegazione è che la loss contrastiva spinge $\mathbf{z}$ verso rappresentazioni fortemente invarianti, sacrificando informazione utile per task diversi; il projection head assorbe questa pressione e protegge $\mathbf{h}$.

Augmentation Ottimale

SimCLR identifica sperimentalmente la pipeline di augmentation ottimale. L'abbinamento più potente è **random cropping + color distortion**:

Pipeline SimCLR — Augmentation

Per ogni immagine, campionare casualmente nell'ordine:

- Random resized crop + horizontal flip
- Color jitter (brightness, contrast, saturation, hue) con probabilità 0.8
- Random grayscale con probabilità 0.2
- Gaussian blur (solo SimCLR v2) con probabilità 0.5

L'augmentation ha una spiegazione teorica precisa: il random crop crea due viste con contesti spaziali diversi ma stesso soggetto semantico, forzando l'encoder a identificare caratteristiche invarianti alla posizione. Il color jitter rende le rappresentazioni invarianti al colore assoluto — senza di esso il modello impara a raggruppare per colore dominante piuttosto che per contenuto. La combinazione è sinergica: ogni trasformazione elimina una diversa scorciatoia che il modello altrimenti sfrutterebbe per minimizzare la loss senza apprendere semantica genuina.

MoCo: Momentum Contrast

119.1 Il Problema del Batch Size e la Soluzione con Queue

Il principale limite pratico di SimCLR è la dipendenza da batch size enormi (4096-8192 immagini) per disporre di sufficienti esempi negativi — un requisito che rende il training costoso in termini di memoria GPU e parallelismo. MoCo (He et al., 2020) aggira questo vincolo con un'architettura più efficiente basata su due innovazioni complementari:

Queue di negativi: invece di usare solo il batch corrente come negativi, si mantiene una queue FIFO di K embedding accumulati dai mini-batch precedenti. La query corrente viene confrontata con tutti i K negativi — il batch effettivo ha dimensione $K = 65536$, indipendente dal batch size B (tipicamente 256). Questo moltiplica i negativi disponibili senza aumentare il consumo di memoria per step.

$$\mathcal{L}_q = -\log \frac{\exp(\mathbf{q} \cdot \mathbf{k}_+ / \tau)}{\exp(\mathbf{q} \cdot \mathbf{k}_+ / \tau) + \sum_{i=1}^{K} \exp(\mathbf{q} \cdot \mathbf{k}_i^- / \tau)}$$

Momentum encoder: per garantire che i negativi in posizioni diverse della queue siano rappresentazioni coerenti tra loro, l'encoder delle key f_k viene aggiornato lentamente come EMA dell'encoder delle query f_q. Con $m = 0.999$, f_k cambia dello 0,1% per step — abbastanza lento da mantenere la coerenza della queue, abbastanza veloce da inseguire i progressi di f_q.

$$\theta_k \leftarrow m\theta_k + (1-m)\theta_q, \qquad m = 0.999$$

Il momentum encoder evolve gradualmente: le chiavi nella queue sono generate da encoder quasi identici in istanti temporali vicini, garantendo coerenza delle rappresentazioni. Se f_k fosse aggiornato con SGD normale, le chiavi in posizioni distanti della queue sarebbero generate da encoder radicalmente diversi — il confronto tra query e negativi diventerebbe incoerente e la loss instabile.

MoCo v2 e v3

MoCo v2 incorpora le due scoperte chiave di SimCLR — il projection head MLP e l'augmentation aggressiva — migliorando significativamente le performance pur mantenendo la memory efficiency della queue. MoCo v3 fa un ulteriore passo: elimina la queue (ormai superflua con ViT e batch size moderati), usa un backbone Vision Transformer e aggiunge il prediction head e lo stop-gradient di BYOL, segnando la convergenza verso un framework unificato di self-supervised learning.

BYOL: Bootstrap Your Own Latent — Senza Negativi

120.1 Architettura e Meccanismo

BYOL (Grill et al., 2020) capovolta un assunto fondamentale del contrastive learning: è possibile apprendere rappresentazioni di alta qualità *senza* esempi negativi. Questa scoperta è sorprendente perché, in assenza di negativi, la soluzione banale è imparare a proiettare tutto sullo stesso punto nello spazio delle rappresentazioni (mode collapse). BYOL evita questo collasso attraverso un'asimmetria architetturale deliberata. La rete **online** $f_\theta, g_\theta, q_\theta$ (encoder + projector + predictor) prevede le rappresentazioni della rete **target** f_ξ, g_ξ (encoder + projector, senza predictor, aggiornata con EMA):

$$\mathbf{z}_{\text{online}} = g_\theta(f_\theta(\mathbf{x}_1)), \quad \mathbf{p}_{\text{online}} = q_\theta(\mathbf{z}_{\text{online}})$$

$$\mathbf{z}_{\text{target}} = g_\xi(f_\xi(\mathbf{x}_2)), \quad \text{stop-gradient}$$

$$\mathcal{L} = -2\frac{\mathbf{p}_{\text{online}} \cdot \mathbf{z}_{\text{target}}}{\|\mathbf{p}_{\text{online}}\|\|\mathbf{z}_{\text{target}}\|} = \left\| \frac{\mathbf{p}_{\text{online}}}{\|\mathbf{p}_{\text{online}}\|} - \frac{\mathbf{z}_{\text{target}}}{\|\mathbf{z}_{\text{target}}\|} \right\|^2 - 2$$

La loss è simmetrica — si calcola sia nella direzione online→target che target→online — ma è minimizzata *esclusivamente* rispetto a θ, con un esplicito stop-gradient applicato alle rappresentazioni target. La rete target non riceve mai gradienti diretti: evolve solo tramite l'aggiornamento EMA $\xi \leftarrow \tau\xi + (1-\tau)\theta$ con $\tau = 0.996$. Questa asimmetria è la chiave che previene il collasso.

Perché non collassa?

La domanda teorica cruciale è: senza negativi, perché la rete non collassa alla soluzione costante $\mathbf{z} = \mathbf{c}$? L'analisi di Tian et al. (2021) mostra che il collapse è evitato dall'asimmetria architetturale: il predictor q_θ — presente solo nella rete online — rompe la simmetria del gioco di ottimizzazione. La rete online deve predire le rappresentazioni target attraverso una trasformazione appresa, non può semplicemente eguagliare il proprio output; il target si muove lentamente e in modo coerente grazie all'EMA, fornendo un segnale stabile senza degenerare in soluzione banale.

- **Predictor:** q_θ deve predire il target — non può semplicemente copiarlo
- **EMA target:** il target si muove lentamente — crea un "moving target" che impedisce il collapse istantaneo
- **Batch normalization:** normalizza le rappresentazioni implicando diversità nel batch

Gli ablation study confermano che sia il predictor che l'aggiornamento EMA sono componenti necessari: rimuovere il predictor o sostituire l'EMA con una copia fissa porta al collasso immediato delle rappresentazioni. L'interazione tra questi due elementi crea la dinamica stabile che permette l'apprendimento senza negativi.

MAE: Masked Autoencoders

121.1 Architettura e Obiettivo

Il pre-training dei Vision Transformer ha seguito per anni le orme del contrastive learning sviluppato per le CNN: SimCLR, MoCo, DINO costruiscono tutti invarianze tramite coppie di viste augmentate, richiedendo il delicato bilanciamento di esempi positivi e negativi. MAE (Masked Autoencoders, He, Chen, Xie, Li, Dollár e Girshick, 2022 — He avendo anche co-firmato ResNet

e originale ViT) introduce una rottura netta con questa paradigma, ispirandosi invece al successo di BERT nel NLP: invece di costruire coppie positive e negative, si maschera casualmente una frazione molto alta di patch (tipicamente il 75%) e si chiede alla rete di ricostruirle a partire dai soli patch visibili. Il mascheramento aggressivo è una scelta deliberata: patch di immagine adiacenti sono altamente ridondanti (a differenza dei token linguistici), per cui una percentuale bassa sarebbe troppo facile da riempire con interpolazione locale. L'architettura è deliberatamente asimmetrica per massimizzare l'efficienza computazionale. L'architettura è deliberatamente asimmetrica per massimizzare l'efficienza computazionale: un encoder pesante (ViT-Large o ViT-Huge) elabora solo il 25% dei token visibili, mentre un decoder leggero ricostruisce le patch mascherate nello spazio dei pixel.

$$\mathcal{L}_{\text{MAE}} = \frac{1}{|\mathcal{M}|} \sum_{i \in \mathcal{M}} \|\hat{\mathbf{x}}_i - \mathbf{x}_i\|^2$$

dove $\mathcal{M}$ è l'insieme dei patch mascherati e $\mathbf{x}_i$ è il valore del patch normalizzato (media e varianza per patch). Si ricostruisce ogni pixel, non una rappresentazione latente.

La masking ratio del 75% è cruciale e non ovvia: con mascherature basse (25–50%), il task è troppo facile e il modello apprende scorciatoie spaziali locali — interpola i pixel vicini invece di capire il contenuto semantico. Con il 75% mascherato, per ricostruire correttamente una patch è necessario comprendere l'oggetto nella sua interezza, spingendo il modello verso rappresentazioni semantiche di alto livello.

Efficienza Computazionale

L'encoder ViT elabora solo il 25% dei token (patch visibili) — riduce il costo quadratico dell'attenzione di 4×. I token [MASK] vengono aggiunti solo nel decoder leggero. Questo permette pre-training su dataset grandi in tempo ragionevole.

DINO: Self-Distillation con No Labels

122.1 Knowledge Distillation Senza Supervisore Esterno

DINO (Caron et al., 2021) introduce una forma di self-distillation in cui sia student che teacher sono la stessa rete in momenti diversi del training. Il teacher è l'EMA dello student — nessun supervisore esterno, nessuna label — eppure emergono spontaneamente rappresentazioni con una qualità semantica straordinaria. Il teacher processa global crops (almeno il 50% dell'immagine, per avere contesto globale), mentre lo student vede sia global che local crops (piccole finestre).

La loss è una cross-entropy tra le distribuzioni di probabilità delle due reti:

$$\mathcal{L}_{\text{DINO}} = -\sum_{\mathbf{x}\in\{g_1,g_2\}}\sum_{\mathbf{x}'\neq\mathbf{x}} P_t(\mathbf{x})\log P_s(\mathbf{x}')$$

dove $P_t(\mathbf{x}) = \text{softmax}(h_t(\mathbf{x})/\tau_t)$ e $P_s(\mathbf{x}') = \text{softmax}(h_s(\mathbf{x}')/\tau_s)$ con $\tau_t < \tau_s$. La temperatura più bassa del teacher ($\tau_t \approx 0.04$ vs $\tau_s \approx 0.1$) rende le sue distribuzioni più "sharp" — il teacher produce assegnamenti di classe più decisi, che fungono da supervisione soft di alta qualità per lo student.

Centering: prevenzione del collapse

DINO usa il **centering** per prevenire che il teacher collassi a un singolo mode (output costante). Il centro $\mathbf{c}$ è aggiornato con EMA dell'output del teacher:

$$\mathbf{c} \leftarrow m\mathbf{c} + (1-m)\frac{1}{B}\sum_{i=1}^{B} g_t(\mathbf{x}_i^{(1)})$$

$$P_t(\mathbf{x}) = \text{softmax}\left(\frac{g_t(\mathbf{x}) - \mathbf{c}}{\tau_t}\right)$$

Sottraendo il centro dall'output del teacher, nessun neurone di output può dominare — il collapse alla soluzione costante è impedito. Il centering è applicato solo al teacher; lo student è regolarizzato implicitamente tramite la sharpening dello student (temperatura alta $\tau_s = 0.1$).

Proprietà emergenti: segmentazione implicita

Una delle scoperte più affascinanti di DINO è la segmentazione semantica emergente: i feature del token [CLS] mostrano, senza alcuna supervisione esplicita, una localizzazione degli oggetti di qualità comparabile a metodi supervisionati. La self-attention del ViT si focalizza naturalmente sulle regioni semanticamente rilevanti dell'immagine, producendo mappe di attenzione che delineano con precisione i contorni degli oggetti. DINOv2 scala questo approccio a dataset curati di 142M immagini e backbone più grandi, ottenendo feature universali di qualità eccezionale.

Setup Formale del Few-Shot Learning

123.1 Training Episodico: N-way K-shot

Il **few-shot learning** affronta una delle sfide più fondamentali: un bambino esposto a tre foto di un wombat lo riconosce il giorno dopo — i sistemi tradizionali non ci riescono con meno di centinaia di esempi. La soluzione non è accumulare più dati per le nuove classi, ma imparare una capacità di adattamento meta-livello. Il setup standard N-way K-shot simula questo scenario in modo rigoroso e riproducibile, valutando la generalizzazione a classi mai viste dura

- Il modello deve classificare tra N classi nuove
- Dispone di K esempi etichettati per classe (**support set** $\mathcal{S}$, $N \times K$ esempi totali)
- Deve classificare una query set $\mathcal{Q}$ di esempi nuovi

Il **training episodico** simula questo scenario durante il training: ad ogni episodio si campionano N classi e K esempi come support, e il modello viene valutato su query. Questo allinea train e test distribution.

$$\min_{\theta} \mathbb{E}_{\mathcal{T} \sim p(\mathcal{T})}[\mathcal{L}_{\mathcal{T}}(f_\theta, \mathcal{S}_{\mathcal{T}}, \mathcal{Q}_{\mathcal{T}})]$$

Il dataset di training $\mathcal{D}_{\text{train}}$ e quello di test $\mathcal{D}_{\text{test}}$ hanno classi completamente disgiunte: il modello non ha mai visto durante il training le classi su cui viene valutato. Questa separazione è fondamentale — misura la capacità di generalizzazione genuina a nuovi concetti, non la semplice memorizzazione di classi note.

Prototypical Networks

124.1 Derivazione Metrica

Nel panorama del few-shot learning, Prototypical Networks (Snell, Swersky e Zemel, NeurIPS 2017) si ispirano al nearest centroid classifier — uno dei metodi più interpretabili — e lo trasportano nello spazio delle rappresentazioni neurali. L'idea centrale è che ogni classe possa essere rappresentata da un singolo **prototipo** — la media degli embedding del support set. Questa scelta ha un'eleganza teorica e pratica notevole: è computazionalmente banale, interpretabile geometricamente come il baricentro della nuvola di punti della classe, e sorprendentemente efficace anche con $K = 1$ o $K = 5$ esempi:

$$\mathbf{c}_n = \frac{1}{K} \sum_{(\mathbf{x}_i, y_i) \in \mathcal{S}: y_i = n} f_\theta(\mathbf{x}_i)$$

La classificazione di una query $\mathbf{x}$ usa la distanza euclidea ai prototipi, trasformata in una distribuzione di probabilità tramite softmax negativa della distanza. L'encoder f_ϕ viene addestrato episodicamente su queste predizioni: impara rappresentazioni in cui esempi della stessa classe si raggruppano naturalmente attorno al prototipo della classe, e classi diverse sono ben separate nello spazio metrico.

$$P(y = n|\mathbf{x}) = \frac{\exp(-d(f_\theta(\mathbf{x}), \mathbf{c}_n))}{\sum_{n'} \exp(-d(f_\theta(\mathbf{x}), \mathbf{c}_{n'}))}$$

$$\mathcal{L}_{\text{proto}} = -\frac{1}{|\mathcal{Q}|} \sum_{(\mathbf{x}, y) \in \mathcal{Q}} \log P(y|\mathbf{x})$$

Le Prototypical Networks hanno una bellissima interpretazione probabilistica: con distanza euclidea, sono equivalenti a un Mixture of Gaussians con covarianza sfericale identica per tutte le classi. Minimizzare la loss cross-entropy equivale a trovare le medie delle gaussiane (i prototipi) che massimizzano la verosimiglianza dei dati sotto questo modello. Con distanza coseno invece di euclidea, si ottengono performance spesso migliori su spazi di alta dimensione.

MAML: Model-Agnostic Meta-Learning

125.1 L'Idea: Inizializzazione Ottimale per l'Adattamento

MAML (Finn et al., 2017) propone una prospettiva rivoluzionaria sul meta-learning: invece di progettare un algoritmo di apprendimento specializzato, si cerca direttamente un'**inizializzazione ottimale** θ dei parametri della rete. L'intuizione è che da questa inizializzazione, pochi step di gradient descent su qualsiasi nuovo task portino rapidamente a buone performance — l'inizializzazione encode la "conoscenza di come imparare".

$$\min_\theta \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}\left(f_{\theta_i'}\right)$$

$$\text{dove: } \theta_i' = \theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$$

L'outer loop ottimizza θ; l'inner loop esegue K_{inner} step di gradient descent per il task specifico.

Derivazione del meta-gradiente (second-order)

Il gradiente di MAML rispetto a θ richiede di differenziare attraverso l'intero inner loop — una forma di differenziazione del secondo ordine che tiene conto di come ogni passo di gradiente modifica l'effetto dei passi successivi. Questa è la caratteristica che rende MAML computazionalmente oneroso ma teoricamente elegante:

$\nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_{\theta_i'}) = \nabla_{\theta_i'} \mathcal{L}_{\mathcal{T}_i} \cdot \frac{\partial \theta_i'}{\partial \theta}$ Chain rule

$\frac{\partial \theta_i'}{\partial \theta} = \mathbf{I} - \alpha \nabla_\theta^2 \mathcal{L}_{\mathcal{T}_i}(f_\theta)$ Derivando $\theta' = \theta - \alpha \nabla \mathcal{L}$: la derivata di θ è $\mathbf{I}$, quella di $\alpha \nabla \mathcal{L}$ è $\alpha \mathbf{H}$

$\nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_{\theta_i'}) = (\mathbf{I} - \alpha \mathbf{H}_i)^T \nabla_{\theta_i'} \mathcal{L}_{\mathcal{T}_i}$ Sostituendo

Il termine $\mathbf{H}_i = \nabla_\theta^2 \mathcal{L}_{\mathcal{T}_i}$ è la Hessiana della loss sul task i, che richiede differenziazione del secondo ordine. Il costo computazionale è $O(|\theta|^2)$ in memoria e $O(|\theta|^2 K_{\text{inner}})$ in tempo — proibitivo per reti grandi. First-Order MAML (FOMAML) ignora questo termine, usando solo il gradiente del primo ordine: sorprendentemente, le performance degradano poco in pratica, suggerendo che il termine Hessiano contribuisce poco all'aggiornamento efficace.

Reptile: Approssimazione First-Order di MAML

126.1 Derivazione e Motivazione

Reptile (Nichol, Achiam e Schulman, OpenAI 2018) nasce da una domanda pratica e pressante: MAML ha proprietà teoriche eleganti e funziona bene empiricamente, ma il suo costo computazionale — dovuto alla differenziazione attraverso l'inner loop — lo rende proibitivo per architetture profonde e task complessi. È possibile ottenere le stesse proprietà di meta-learning senza doppia differenziazione? Reptile risponde affermativamente, semplificando ulteriormente MAML ed eliminando completamente la struttura bilevel: esegue semplicemente k step di SGD sul task $\mathcal{T}_i$ partendo da θ e poi sposta θ nella direzione dei parametri finali ottenuti. L'assenza di doppia differenziazione rende Reptile computazionalmente leggero quanto il fine-tuning standard, ma con proprietà di meta-learning sorprendenti:

$$\theta_i^{(k)} = \text{SGD}^k(\theta, \mathcal{T}_i) \quad \text{(k step di SGD su task } i\text{)}$$

$$\theta \leftarrow \theta + \varepsilon(\theta_i^{(k)} - \theta)$$

Reptile può essere analizzato formalmente tramite Taylor expansion al secondo ordine dell'aggiornamento, rivelando che ottimizza implicitamente un obiettivo meta-learning:

$$\varepsilon(\theta_i^{(k)} - \theta) \approx -\varepsilon k\alpha \nabla \mathcal{L}_{\mathcal{T}_i}(\theta) + \varepsilon k\alpha^2 \sum_{j<k} \mathbf{H}_j \nabla \mathcal{L}_{\mathcal{T}_i}$$

Il primo termine è un gradient descent diretto sulla loss media del task — come nel fine-tuning standard. Il secondo termine, la novità di Reptile, è la correzione del secondo ordine: Reptile ottimizza implicitamente per la curvatura della loss, cercando regioni dello spazio dei parametri in cui i gradienti di task diversi sono allineati — esattamente ciò che rende un'inizializzazione buona per il meta-learning.

Meta-SGD e ANIL

127.1 Meta-SGD: Learning Rate come Parametro

Meta-SGD (Li et al., 2017) estende MAML in una direzione naturale: invece di usare un unico learning rate scalare per tutti i parametri, rende il learning rate $\boldsymbol{\alpha}$ un vettore learnable con un valore per ogni parametro della rete. Questo permette al meta-learner di scoprire automaticamente quali parametri devono adattarsi rapidamente a nuovi task e quali devono rimanere stabili:

$$\theta_i' = \theta - \boldsymbol{\alpha} \odot \nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$$

Il meta-gradiente rispetto a $\boldsymbol{\alpha}$: $\nabla_{\boldsymbol{\alpha}} \mathcal{L}_{\mathcal{T}_i}(f_{\theta_i'}) = -\nabla_{\theta_i'} \mathcal{L}_{\mathcal{T}_i} \odot \nabla_\theta \mathcal{L}_{\mathcal{T}_i}$ (hadamard del gradiente outer con il gradiente inner). Meta-SGD impara learning rate adattativi per-parametro — notevolmente più potente di MAML con learning rate scalare.

ANIL: Almost No Inner Loop

ANIL (Almost No Inner Loop, Raghu et al., 2020) parte da un'osservazione empirica sorprendente: il vantaggio di MAML viene quasi interamente dall'adattamento dell'ultimo layer (la testa di classificazione), non dal corpo profondo della rete. Le feature estratte dagli strati profondi sono già sufficientemente generali da trasferirsi bene ai nuovi task senza aggiornamento. ANIL sfrutta questa osservazione adattando *solo* la testa nell'inner loop:

$$\theta'_{\text{head}} = \theta_{\text{head}} - \alpha \nabla_{\theta_{\text{head}}} \mathcal{L}_{\mathcal{T}_i}$$

$$\theta'_{\text{body}} = \theta_{\text{body}} \quad \text{(nessun inner update)}$$

ANIL risulta circa 2× più veloce di MAML a parità di performance — la maggior parte del calcolo in MAML è sprecata nell'adattare strati profondi che non cambiano significativamente tra task. Questo risultato ha importanti implicazioni teoriche: suggerisce che il valore del meta-learning è meno nell'adattamento rapido della feature extractor e più nell'apprendimento di una buona inizializzazione per la testa di classificazione.

Barlow Twins, VICReg e Analisi Teorica dell'Invarianza

122b.1 Barlow Twins: Ridondanza Ridotta

Barlow Twins (Zbontar et al., 2021) usa la **cross-correlation matrix** tra le due viste come obiettivo. Date le rappresentazioni normalizzate $\mathbf{Z}^A, \mathbf{Z}^B \in \mathbb{R}^{N \times d}$ (batch × dim):

$$\mathcal{C}_{ij} = \frac{\sum_n Z^A_{n,i} Z^B_{n,j}}{\sqrt{\sum_n (Z^A_{n,i})^2} \sqrt{\sum_n (Z^B_{n,j})^2}}$$

$$\mathcal{L}_{\text{BT}} = \underbrace{\sum_i (1 - \mathcal{C}_{ii})^2}_{\text{invarianza}} + \lambda \underbrace{\sum_i \sum_{j \neq i} \mathcal{C}_{ij}^2}_{\text{ridondanza ridotta}}$$

Il termine diagonale spinge $\mathcal{C}_{ii} \to 1$ — le rappresentazioni delle due viste devono essere invarianti all'augmentation. Il termine off-diagonale spinge $\mathcal{C}_{ij} \to 0$ per $i \neq j$ — le diverse dimensioni

della rappresentazione devono essere decorrelate (non ridondanti). Barlow Twins non richiede né negativi né stop-gradient — il decorrelation penalty implicitamente previene il collapse.

VICReg: Variance-Invariance-Covariance Regularization

VICReg (Bardes et al., 2022) rende esplicite le tre componenti necessarie per evitare il collapse:

$$\mathcal{L}_{\text{VICReg}} = \lambda s(\mathbf{Z}, \mathbf{Z}') + \mu v(\mathbf{Z}, \mathbf{Z}') + \nu c(\mathbf{Z}, \mathbf{Z}')$$

- **Invariance:** $s(\mathbf{Z}, \mathbf{Z}') = \frac{1}{N}\sum_i \|z_i - z_i'\|^2$ — minimizza la distanza tra le due viste
- **Variance:** $v(\mathbf{Z}) = \frac{1}{d}\sum_j \max(0, \gamma - \text{std}(z_j))$ — mantiene la varianza di ogni dimensione sopra γ (previene il collapse a costante)
- **Covariance:** $c(\mathbf{Z}) = \frac{1}{d}\sum_{i \neq j}[\text{Cov}(Z)_{ij}]^2$ — riduce la covarianza off-diagonale (come Barlow Twins)

122b.2 Analisi Teorica: Perché l'Augmentation Funziona

Tian et al. (2020) dimostrano che la qualità delle rappresentazioni apprese dipende criticamente dall'**augmentation graph**: un grafo su cui i campioni dello stesso dato formano una componente connessa tramite le augmentation. Il contrastive learning impara a classificare in base alle componenti connesse di questo grafo.

Il **conditional entropy bound**: la mutual information tra la rappresentazione $\mathbf{z}$ e il label y è limitata dall'informazione mutua tra le due viste:

$$I(\mathbf{z}; y) \leq I(\mathbf{z}; \mathbf{z}') \leq I(\mathbf{z}; v_1) \leq I(\mathbf{x}; \mathbf{x}') = H(\mathbf{x}') - H(\mathbf{x}'|\mathbf{x})$$

Il termine $H(\mathbf{x}'|\mathbf{x})$ è la "rumorosità" dell'augmentation — augmentation troppo aggressive rimuovono informazione semantica e peggiorano le performance a valle.

122b.3 Self-Supervised per Testo: SimCSE

SimCSE (Gao et al., 2021) applica contrastive learning alle sentence embeddings. La chiave: usare il **dropout come augmentation minima**. La stessa frase viene passata due volte allo

stesso encoder (con dropout distinto ogni volta) — le due rappresentazioni con dropout diverso formano la coppia positiva:

$$h_i = f_\theta(\mathbf{x}_i, \varepsilon^{(1)}), \quad h_i^+ = f_\theta(\mathbf{x}_i, \varepsilon^{(2)})$$

$$\mathcal{L}_{\text{SimCSE}} = -\log \frac{\exp(\text{sim}(h_i, h_i^+)/\tau)}{\sum_{j=1}^{N} \exp(\text{sim}(h_i, h_j^+)/\tau)}$$

Risultato sorprendente: questo "minimo intervento" supera metodi di augmentation testuale molto più elaborati (paraphrasing, back-translation, word deletion). La spiegazione: l'augmentation testuale può alterare il significato o creare coppie positive di scarsa qualità — il dropout è il "minimo rumore" che preserva la semantica.

Meta-Learning Bayesiano e Algoritmo di Ottimizzazione delle Policy

127b.1 SNAIL: Attenzione Temporale per Meta-Learning

SNAIL (Mishra et al., 2018) usa una combinazione di convoluzioni causali temporali e attenzione per il meta-learning. L'idea: dato il support set come sequenza di esempio-label $(x_1, y_1, \ldots, x_K, y_K, x_{\text{query}})$, il modello elabora questa sequenza e predice y_{query}:

$$\mathbf{h}_t = \text{TC-Block}(\mathbf{h}_{1:t}) + \text{Attention-Block}(\mathbf{h}_{1:t}, \mathbf{h}_t)$$

I **Temporal Convolutional blocks** aggregano informazione locale in modo denso; i **Causal Attention blocks** permettono di integrare qualsiasi coppia passata con attenzione selettiva. SNAIL può "ricordare" selettivamente gli esempi del support set più rilevanti per la query corrente.

127b.2 Meta-Learning come Inferenza Bayesiana

La prospettiva bayesiana del meta-learning: la distribuzione a priori sui task $p(\mathcal{T})$ e la posterior sui parametri del task $p(\theta|\mathcal{S}_\mathcal{T})$ definiscono un framework unificato. MAML può essere visto come MAP inference: $\theta_i' = \arg\max_\theta \log p(\mathcal{S}_i|\theta)p(\theta|\theta_{\text{meta}})$ con prior gaussiano $p(\theta|\theta_{\text{meta}}) = \mathcal{N}(\theta_{\text{meta}}, \mathbf{I}/\alpha)$.

VERSA (Gordon et al., 2019) usa un encoder amortizzato $q_\phi(\mathbf{z}|\mathcal{S})$ per inferire direttamente la posterior del task:

$$p(y|\mathbf{x}, \mathcal{S}) = \int p_\theta(y|\mathbf{x}, \mathbf{z}) q_\phi(\mathbf{z}|\mathcal{S})\, d\mathbf{z} \approx \sum_{k=1}^{K} p_\theta(y|\mathbf{x}, \mathbf{z}^{(k)})$$

La predizione è una media di ensemble su campioni $\mathbf{z}^{(k)} \sim q_\phi(\mathbf{z}|\mathcal{S})$ — cattura l'incertezza del task. Il training è ELBO: massimizzare $\mathbb{E}_{q_\phi}[\log p(y|\mathbf{x}, \mathbf{z})] - D_{\mathrm{KL}}(q_\phi \| p(\mathbf{z}))$.

127b.3 Confronto dei Metodi di Meta-Learning

Metodo	Tipo	Inner loop	Memoria	Punti di forza
Prototypical Nets	Metric-based	No	$O(NK)$	Semplicità, velocità
Matching Nets	Metric-based	No	$O(NK)$	Full context embedding
MAML	Optimization-based	Sì (k step)	$O(\|\theta\|)$	Generalità, flessibilità
Reptile	Optimization-based	Sì (k step)	$O(\|\theta\|)$	Efficienza, first-order
SNAIL	Model-based	No	$O(NK \cdot d)$	Memoria contestuale flessibile
VERSA	Bayesian	No (amortizzato)	$O(\|\phi\|)$	Incertezza sul task

VC Dimension: Shattering, Growth Function e Lemma di Sauer

128.1 Shattering e VC Dimension

La VC dimension fornisce la misura più fondamentale della complessità di una classe di ipotesi, con un'eleganza matematica che ha ispirato decenni di ricerca sulla generalizzazione. Sia $\mathcal{H}$ una classe di funzioni $h : \mathcal{X} \to \{0, 1\}$. Un insieme $C = \{x_1, \ldots, x_m\} \subset \mathcal{X}$ è **shattered** da $\mathcal{H}$ se per ogni possibile etichettatura binaria di C esiste una funzione in $\mathcal{H}$ che la realizza — la classe è abbastanza espressiva da separare i punti in qualsiasi modo:

Definizione — VC Dimension

La VC dimension di $\mathcal{H}$ è la cardinalità del più grande insieme che $\mathcal{H}$ riesce a shatterare — se $\mathcal{H}$ può shatterare insiemi di qualsiasi dimensione, si dice che $d_{\mathrm{VC}}(\mathcal{H}) = \infty$:

$$d_{\mathrm{VC}}(\mathcal{H}) = \max\{m : \exists C \subset \mathcal{X}, |C| = m, C \text{ shattered da } \mathcal{H}\}$$

Esempi:

Iperspazio $\mathcal{H}$	d_{VC}	**Intuzione**
Semispazi in $\mathbb{R}^d$: $\text{sign}(\mathbf{w}^T\mathbf{x}+b)$	$d+1$	$d+1$ punti in posizione generale
Intervalli $[a,b]$ in $\mathbb{R}$	2	2 punti separati, non 3
Neuroni con sigmoide in $\mathbb{R}^d$	$d+1$	Come i semispazi asintoticamente
MLP: L layer, W pesi totali	$O(WL\log W)$	Bartlett 1998
Funzioni costanti $h(x)=c$	0	Nessun insieme può essere shattered

128.2 Lemma di Sauer-Shelah e Growth Function

La **growth function** $\Pi_{\mathcal{H}}(m)$ conta il massimo numero di dicotomie (etichettature binarie distinte) che $\mathcal{H}$ può produrre su m punti:

$$\Pi_{\mathcal{H}}(m) = \max_{C\subset\mathcal{X},|C|=m} |\{(h(x_1),\dots,h(x_m)) : h\in\mathcal{H}\}|$$

Lemma di Sauer-Shelah

Se $d_{\text{VC}}(\mathcal{H}) = d < \infty$, allora per ogni $m \geq d$:

$$\Pi_{\mathcal{H}}(m) \leq \sum_{i=0}^{d}\binom{m}{i} \leq \left(\frac{em}{d}\right)^d$$

In particolare, non appena $m > d$, la growth function è polinomiale in m invece che esponenziale 2^m. Questo è il salto qualitativo fondamentale: una classe con VC dimension finita ha una complessità "governabile" — non può shatterare insiemi arbitrariamente grandi — e questo si traduce in garanzie di generalizzazione non banali.

Prova per induzione su $m+d$. Il caso base: $m=0$ o $d=0$ è triviale. Per il passo induttivo: partizioniamo l'insieme C di m punti in un punto x_m e $C' = C\setminus\{x_m\}$. Le dicotomie di $\mathcal{H}|_C$ sono al più le dicotomie di $\mathcal{H}|_{C'}$ più le dicotomie dove togliere x_m produce un duplicato — e queste ultime corrispondono a dicotomie di una classe con $d_{\text{VC}} \leq d-1$ su $m-1$ punti.

PAC Learning: Sample Complexity Rigorosa

129.1 Framework PAC

Il framework PAC (Probably Approximately Correct, Valiant, 1984) ha formalmente risolto la domanda "quando è possibile apprendere?", dando una risposta matematicamente precisa che ha fondato la teoria computazionale dell'apprendimento. Una classe $\mathcal{H}$ è PAC-learnable se esiste un algoritmo che, con probabilità almeno $1-\delta$ e con errore al più ε, trova un ipotesi corretta usando un numero polinomiale di campioni:

$$P_{S\sim\mathcal{D}^m}\left[R(A(S)) - \min_{h\in\mathcal{H}} R(h) \leq \varepsilon\right] \geq 1-\delta$$

Affinché questo valga per $m \geq m_{\mathcal{H}}(\varepsilon,\delta)$ campioni sufficienti. Il teorema fondamentale del PAC learning stabilisce che $\mathcal{H}$ è PAC-learnable se e solo se $d_{\text{VC}}(\mathcal{H}) < \infty$: la VC dimension finita è sia condizione necessaria che sufficiente per l'apprendibilità. Questo risultato unifica la complessità statistica con la complessità computazionale in modo profondo.

Sample Complexity con VC Dimension

Dalla disuguaglianza di Sauer e dalla disuguaglianza di Hoeffding (tramite la disuguaglianza di Chernoff-Hoeffding):

$$P\left(\sup_{h\in\mathcal{H}} |R(h) - \hat{R}_m(h)| > \varepsilon\right) \leq 4\Pi_{\mathcal{H}}(2m)e^{-m\varepsilon^2/8} \leq 4\left(\frac{2em}{d}\right)^d e^{-m\varepsilon^2/8}$$

Ponendo questo $\leq \delta$ e risolvendo per m:

$$m_{\mathcal{H}}(\varepsilon,\delta) = O\left(\frac{d\log(1/\varepsilon) + \log(1/\delta)}{\varepsilon^2}\right)$$

Il numero di campioni necessari cresce linearmente con la VC dimension — classi più espressive richiedono più dati per apprendere in modo affidabile. Il teorema è sia upper bound (questo numero di campioni è sufficiente) che lower bound (meno campioni non bastano in generale): $\Omega\left(\frac{d+\log(1/\delta)}{\varepsilon}\right) \leq m_{\mathcal{H}}(\varepsilon,\delta) \leq O\left(\frac{d\log(d/\varepsilon)+\log(1/\delta)}{\varepsilon}\right)$. La dipendenza logaritmica da $1/\delta$ mostra che aumentare la confidenza è molto più economico che ridurre l'errore.

Rademacher Complexity: Bound Tight sulla Generalizzazione

130.1 Definizione e Bound

La **Rademacher complexity** misura la capacità di $\mathcal{H}$ di correlare con rumore casuale — più è alta, più la classe è espressiva.

Definizione — Rademacher Complexity Empirica

$$\hat{\mathfrak{R}}_S(\mathcal{H}) = \mathbb{E}_{\boldsymbol{\sigma}}\left[\sup_{h\in\mathcal{H}} \frac{1}{m}\sum_{i=1}^{m} \sigma_i h(x_i)\right]$$

dove $\sigma_i \sim$ Rademacher(± 1) uniformemente. La Rademacher complexity (non empirica) è $\mathfrak{R}_m(\mathcal{H}) = \mathbb{E}_S[\hat{\mathfrak{R}}_S(\mathcal{H})]$.

Teorema (Koltchinskii-Panchenko, 2002)

Per ogni $\delta > 0$, con probabilità $\geq 1-\delta$ su $S \sim \mathcal{D}^m$:

$$R(h) \leq \hat{R}_m(h) + 2\mathfrak{R}_m(\mathcal{H}) + \sqrt{\frac{\ln(1/\delta)}{2m}} \quad \forall h \in \mathcal{H}$$

Calcolo della Rademacher per classi specifiche

Per la classe di funzioni lineari $\mathcal{H}_B = \{\mathbf{x} \mapsto \mathbf{w}^T\mathbf{x} : \|\mathbf{w}\| \leq B\}$ con dati limitati $\|\mathbf{x}_i\| \leq R$, la Rademacher complexity si calcola esplicitamente tramite la dualità di Cauchy-Schwarz, ottenendo un bound che dipende dal prodotto BR (margin-normalized complexity) ma non dalla dimensione dello spazio di input d:

$$\mathfrak{R}_m(\mathcal{H}_B) = \mathbb{E}_{\boldsymbol{\sigma}}\left[\sup_{\|\mathbf{w}\|\leq B} \frac{\mathbf{w}^T}{m}\sum_{i=1}^{m}\sigma_i\mathbf{x}_i\right] = \frac{B}{m}\mathbb{E}_{\boldsymbol{\sigma}}\left[\left\|\sum_{i=1}^{m}\sigma_i\mathbf{x}_i\right\|\right] \leq \frac{B\sqrt{\mathbb{E}[\|\sum\sigma_i\mathbf{x}_i\|^2]}}{m} = \frac{BR\sqrt{m}}{m} = \frac{BR}{\sqrt{m}}$$

Il bound sulla generalizzazione per $\mathcal{H}_B$ diventa: $R(h) \leq \hat{R}_m(h) + 2BR/\sqrt{m} + O(\sqrt{\log(1/\delta)/m})$ — notevolmente, è indipendente dalla dimensione dell'input d (ma dipende dalla norma B dei pesi e dal raggio R dei dati). Questo spiega perché la regolarizzazione L2 (che limita B) migliora la generalizzazione anche quando il modello è sovrappararmetrizzato: non è la dimensione ma la norma che conta.

Confronto Rademacher vs VC

I bound basati su Rademacher complexity sono tipicamente più tight di quelli basati su VC dimension perché sfruttano la struttura specifica del dataset osservato (tramite $\hat{\mathfrak{R}}_S$) invece di considerare il caso peggiore su tutti i possibili dataset, e perché la dipendenza dalla distribuzione dei dati è catturata in modo più granulare attraverso il prodotto $\langle \boldsymbol{\sigma}, \mathbf{h}(S) \rangle$.

- Sono data-dependent (la complexity empirica $\hat{\mathfrak{R}}_S$ si calcola sul dataset specifico)
- Catturano la geometria della distribuzione dei dati, non solo la complessità worst-case della classe
- Si adattano naturalmente a classi di funzioni reali (non solo classificatori binari)

Stabilità Algoritmica e Generalizzazione

131.1 Definizione di Stabilità Uniforme

L'approccio tradizionale alla generalizzazione — basato sulla complessità della classe di ipotesi come VC dimension o Rademacher complexity — ha un limite fondamentale: caratterizza la classe $\mathcal{H}$ nel suo insieme, non il comportamento dell'algoritmo specifico che viene eseguito. Due algoritmi che restituiscono ipotesi dalla stessa classe possono generalizzare in modo molto diverso a seconda della loro sensibilità ai singoli esempi di training. La **stabilità algoritmica**, sviluppata sistematicamente da Bousquet e Elisseeff nel loro lavoro seminale del 2002, cattura esattamente questa proprietà: quanto cambia il predittore prodotto dall'algoritmo se si rimuove o sostituisce un singolo esempio dal dataset? Un algoritmo è stabile se tale perturbazione ha effetto trascurabile, e la stab ilità si traduce direttamente in generalizzazione. Formalmente, un algoritmo A è β**-uniformly stable** se per ogni coppia di dataset S e S^i (identici tranne che per l'i-esimo esempio, che in S^i è sostituito da un esempio diverso):

$$\sup_z |\ell(A(S), z) - \ell(A(S^i), z)| \leq \beta$$

Teorema (Bousquet e Elisseeff, 2002)

Se A è β-uniformly stable e la loss è limitata in $[0, M]$, con probabilità $\geq 1 - \delta$:

$$R(A(S)) \leq \hat{R}_m(A(S)) + \beta + (4m\beta + M)\sqrt{\frac{\ln(1/\delta)}{2m}}$$

Il bound di generalizzazione che ne consegue è sorprendente: dipende solo da β (la stabilità dell'algoritmo) e da M (il bound sulla loss), ma non dalla VC dimension della classe di ipotesi. In particolare, algoritmi molto stabili — come SGD con learning rate che decade — generalizzano bene anche su classi overparametrizzate, il che è una delle chiavi per capire il successo del deep learning.

SGD è stabile

Hardt et al. (2016) dimostrano che SGD con T step e learning rate $\alpha_t \leq c/t$ è β-stabile con $\beta = O(\alpha T/m)$. Quindi, per reti neurali con loss Lipschitz, la generalizzazione è garantita se T/m non cresce troppo — si può imparare bene finché il numero di step non supera linearmente il numero di esempi. Questo risultato giustifica l'uso di early stopping come forma implicita di regolarizzazione e collega la stabilità con la durata del training.

Double Descent: Fenomenologia Rigorosa

132.1 Il Fenomeno

La curva di rischio classica della teoria bias-varianza prevede un unico minimo in corrispondenza del modello "right-sized": modelli troppo semplici hanno alto bias, modelli troppo complessi hanno alta varianza, e il punto ottimale bilancia i due estremi. Il **double descent** (Belkin et al., 2019) documenta un fenomeno che contraddict questa visione: la curva di errore in funzione della capacità del modello ha una forma a doppia discesa, con un picco al punto di interpolazione seguito da una sorprendente riduzione dell'errore nei regimi sovrappararmetrizzati.

$$\text{Errore test} \approx \begin{cases} \text{bias-varianza classico} & N_{\text{param}} < N_{\text{data}} \\ +\infty \text{ (overfitting completo)} & N_{\text{param}} \approx N_{\text{data}} \\ \text{generalizzazione migliore} & N_{\text{param}} \gg N_{\text{data}} \end{cases}$$

Il picco di errore al punto di interpolazione è reale e riproducibile: corrisponde al momento in cui il modello ha esattamente abbastanza parametri per memorizzare i dati di training, ma lo fa in modo "rough" — le soluzioni sono altamente sensibili alle piccole perturbazioni dell'input. Superato questo punto, nel regime sovrappararmetrizzato, esistono molte soluzioni con training error zero e SGD — grazie alla sua struttura implicita di minimo-norma — converge alla più regolare tra queste, che generalizza meglio.

Analisi nel modello lineare

Per regressione lineare con d feature, n campioni e $d > n$, la soluzione a norma minima è $\hat{\mathbf{w}} = \mathbf{X}^{+}\mathbf{y}$ (pseudo-inversa). Il rischio:

$$R(\hat{\mathbf{w}}) = \underbrace{\sigma^2 \frac{n}{d-n-1}}_{\text{varianza (scompare per } d \gg n)} + \underbrace{\|\mathbf{w}^*\|^2 \left(1 - \frac{n}{d}\right)}_{\text{bias (scompare per } d \to \infty)}$$

Per $d \gg n$ entrambi i termini tendono a zero: la soluzione a norma minima generalizza bene perché ha norma piccola, e il termine di varianza decresce grazie all'overparametrizzazione. Il picco di errore si manifesta al punto critico $d \approx n$ dove la matrice $\mathbf{X}^T\mathbf{X}$ è quasi singolare e la norma della soluzione esplode — un momento di transizione di fase analogo ai fenomeni critici in fisica statistica.

Benign Overfitting: Interpolation Threshold

133.1 Quando l'Overfitting è Benigno — Contesto e Motivazione

Per quasi un secolo, l'overfitting è stato considerato il nemico naturale della generalizzazione: un modello che memorizza perfettamente i dati di training sembrava destinato a fallire su dati nuovi. Questa visione, radicata nella teoria classica (VC dimension, bias-variance tradeoff), è stata messa in crisi da un'osservazione empirica sorprendente: reti neurali profondamente sovraparametrizzate che interpolano esattamente i dati di training — incluso il rumore — spesso generalizzano meglio di modelli deliberatamente regolarizzati per evitare l'interpolazione. Bartlett et al. (2020) forniscono la prima spiegazione rigorosa di questo fenomeno, dimostrando che l'overfitting può essere **benigno** — cioè non ostacola la generalizzazione — in condizioni specifiche e precisamente caratterizzabili. Per regressione lineare con dati rumorosi $y_i = \mathbf{x}_i^T\mathbf{w}^* + \varepsilon_i$, la soluzione a norma minima $\hat{\mathbf{w}}$ soddisfa $\hat{\mathbf{w}}^T\mathbf{x}_i = y_i$ esattamente (interpolazione). Il rischio di generalizzazione:

$$R(\hat{\mathbf{w}}) \leq O\left(\frac{\|\mathbf{w}^*\|^2}{\lambda_n}\right) + O\left(\sigma^2 \frac{\sum_{k>n} \lambda_k / \lambda_n^2}{(k_n/\lambda_n)^2}\right)$$

dove $\lambda_1 \geq \lambda_2 \geq \cdots$ sono gli autovalori della matrice di covarianza dell'input e $k_n = \sum_{k>n} \lambda_k / \lambda_n$ è la dimensione effettiva della "coda" della distribuzione.

L'overfitting è benigno quando la distribuzione ha una coda con molte dimensioni di piccola

varianza — l'interpolazione avviene nelle direzioni di bassa energia, che non influenzano la predizione su nuovi dati. Condizione sufficiente: $k_n/n \to \infty$ (il numero di dimensioni effettive cresce più velocemente dei dati).

Data Parallelism, Model Parallelism e Pipeline Parallelism

134.1 Data Parallelism

In **Data Parallelism (DP)**, ogni worker ha una copia del modello e processa un sottoinsieme del batch. Dopo il forward+backward, i gradienti vengono aggregati (AllReduce) e ogni worker aggiorna il proprio modello identicamente.

La comunicazione richiesta: un'operazione **AllReduce** sul gradiente $\nabla_\theta \mathcal{L} \in \mathbb{R}^{|\theta|}$. Con K worker e l'algoritmo ring-AllReduce:

$$\text{Comunicazione totale} = 2(K-1)\frac{|\theta|}{K} \approx 2|\theta| \quad \text{per } K \text{ grande}$$

Ring-AllReduce è ottimale in termini di volume di dati trasmessi, e il costo di comunicazione è praticamente indipendente da K. Il problema principale rimane la memoria: ogni worker deve mantenere in memoria una copia completa del modello, degli optimizer states e delle attivazioni per il backward pass — con modelli da miliardi di parametri, questo è spesso il collo di bottiglia.

134.2 Model Parallelism (Tensor Parallelism)

In **Tensor Parallelism**, ogni worker gestisce una parte dei parametri di ogni layer. Per un layer $\mathbf{Y} = \mathbf{XA}$ con $\mathbf{A} \in \mathbb{R}^{d \times d}$, si partiziona per colonne: $\mathbf{A} = [\mathbf{A}_1 | \mathbf{A}_2 | \cdots | \mathbf{A}_K]$ con $\mathbf{A}_k \in \mathbb{R}^{d \times d/K}$. Worker k calcola $\mathbf{Y}_k = \mathbf{XA}_k$ localmente, poi AllGather aggrega $\mathbf{Y} = [\mathbf{Y}_1 | \cdots | \mathbf{Y}_K]$.

Il costo di comunicazione: 2 operazioni AllReduce/AllGather per layer (forward e backward). Megatron-LM (Shoeybi et al., 2019) implementa questo per i Transformer con 1 AllReduce nel forward e 1 nel backward per ogni attention/FFN block.

134.3 Pipeline Parallelism

In **Pipeline Parallelism**, ogni worker gestisce un gruppo di layer consecutivi. Il problema del "pipeline bubble": mentre worker k esegue il forward di micro-batch i, i worker successivi sono in attesa.

Con M micro-batch e K stage, la frazione di tempo sprecata nel bubble è $\frac{K-1}{M+K-1}$: cresce

con il numero di stage e decresce con il numero di micro-batch. Per minimizzare la bolla, si usano molti micro-batch ($M \gg K$), accettando un leggero aumento del memory overhead per le attivazioni dei micro-batch in-flight.

$$\text{Bubble fraction} = \frac{K-1}{M+K-1} \xrightarrow{M \gg K} 0$$

GPipe (Huang et al., 2019) accumula gradiente su tutti gli M micro-batch prima di aggiornare. PipeDream (Narayanan et al., 2019) usa aggiornamenti asincroni per ridurre ulteriormente il bubble.

ZeRO: Zero Redundancy Optimizer — Analisi Completa

135.1 Analisi della Memoria in Mixed Precision Training

Con l'aumento esponenziale della scala dei modelli — da BERT con 340M parametri a GPT-3 con 175B — il Data Parallelism standard ha rivelato un paradosso di scalabilità: più GPU si aggiungono, maggiore è la memoria sprecata in ridondanza identica. ZeRO (Zero Redundancy Optimizer, Rajbhandari, Rasley, Ruwase e He, Microsoft Research 2020) risolve questo problema distribuendo optimizer states, gradienti e parametri tra i worker invece di replicarli: elimina la ridondanza che è la causa principale dell'inefficienza del Data Parallelism standard, riducendo la memoria per worker da $O(\Psi)$ a $O(\Psi/K)$.

Componente	**Bytes per parametro**	**Totale per Ψ parametri**
Parametri FP16	2	2Ψ
Gradienti FP16	2	2Ψ
Parametri master FP32	4	4Ψ
Adam: momento primo FP32	4	4Ψ
Adam: momento secondo FP32	4	4Ψ
Totale	**16**	16Ψ

ZeRO Stage 1, 2, 3

ZeRO (Rajbhandari et al., 2020) partiziona gli stati dell'optimizer, i gradienti e i parametri tra i K worker:

$$\text{Stage 1: Optimizer state sharding} \quad \rightarrow \quad \text{memoria/worker} = 4\Psi + 12\Psi/K$$

$$\text{Stage 2: + Gradient sharding} \quad \rightarrow \quad \text{memoria/worker} = 4\Psi + 2\Psi/K + 12\Psi/K$$

$$\text{Stage 3: + Parameter sharding} \quad \rightarrow \quad \text{memoria/worker} = 16\Psi/K$$

ZeRO Stage 3 riduce la memoria per worker di un fattore K: con 64 GPU si possono addestrare modelli 64 volte più grandi rispetto al DP standard. Il prezzo è una comunicazione aggiuntiva (AllGather per raccogliere i parametri necessari prima del forward, ReduceScatter per i gradienti dopo il backward), ma con interconnessioni ad alta velocità (NVLink, InfiniBand) questo overhead è accettabile e il throughput rimane elevato.

Quantizzazione: PTQ, QAT e GPTQ

136.1 Post-Training Quantization (PTQ)

La **quantizzazione** riduce la precisione dei pesi da FP32/FP16 a INT8/INT4, ottenendo risparmio di memoria e accelerazione dell'inferenza a spese di una leggera perdita di accuratezza. La quantizzazione uniforme mappa valori in virgola mobile su un reticolo discreto: dato un intervallo $[\alpha, \beta]$ da quantizzare con $N = 2^b$ livelli, la funzione di quantizzazione è:

$$Q(x) = \text{clamp}\left(\text{round}\left(\frac{x - z_{\text{point}}}{s}\right), 0, 2^b - 1\right) \cdot s + z_{\text{point}}$$

dove $s = (\beta - \alpha)/(2^b - 1)$ è lo scale factor e z_{point} è il zero-point. Il **quantization error** è $\Delta x = Q(x) - x$, limitato in $|s|/2$. La scelta di $[\alpha, \beta]$ è cruciale: con clip troppo stretto si perde informazione nei valori grandi (clipping error); con clip troppo largo, la risoluzione per valori tipici è scarsa (rounding error).

Quantization-Aware Training (QAT)

Il Quantization-Aware Training (QAT) simula la quantizzazione durante il training inserendo operazioni di quantizzazione-dequantizzazione nel grafo computazionale. Il problema è che $Q(x)$ non è differenziabile: si usa lo **Straight-Through Estimator** (STE), che nel forward pass usa

$Q(x)$ (il valore quantizzato) e nel backward pass copia il gradiente come se la quantizzazione fosse l'identità. Questa approssimazione, per quanto rozza, funziona sorprendentemente bene in pratica:

$$\text{forward: } \tilde{w} = Q(w), \quad \text{backward: } \frac{\partial \mathcal{L}}{\partial w} = \frac{\partial \mathcal{L}}{\partial \tilde{w}}$$

La rete apprende pesi che rimangono vicini ai punti di quantizzazione — la loss post-quantizzazione migliora significativamente rispetto a PTQ.

GPTQ: Quantizzazione Ottimale per LLM

GPTQ (Frantar et al., 2023) è un algoritmo di PTQ ottimale per LLM che quantizza i pesi di ogni layer minimizzando l'errore quadratico pesato sulla Hessiana del layer. L'idea chiave è che non tutti gli errori di quantizzazione sono uguali: un errore su un peso ad alta curvatura (grande Hessiana) degrada molto più la loss di un errore su un peso a bassa curvatura. Per il layer $\mathbf{W} \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$:

$$\|\mathbf{W}\mathbf{X} - \hat{\mathbf{W}}\mathbf{X}\|_F^2 = \|\Delta\mathbf{W}\mathbf{X}\|_F^2 = \text{tr}(\Delta\mathbf{W}\mathbf{H}\Delta\mathbf{W}^T)$$

dove $\mathbf{H} = \mathbf{X}\mathbf{X}^T$ è la matrice di Hessiana (Gram matrix degli input al layer). GPTQ quantizza colonna per colonna usando aggiornamenti di Cholesky per mantenere la Hessiana aggiornata efficacemente, correggendo gli errori sulle colonne rimanenti tramite la Optimal Brain Surgeon (OBS) formula. In pratica, GPTQ quantizza LLM da 175B parametri a 4-bit in poche ore su una singola GPU, con perdita di accuratezza minima rispetto al modello FP16.

Pruning Strutturato e Knowledge Distillation

137.1 Pruning: Magnitudine e L0 Sparsification

Il **pruning** riduce la dimensione della rete rimuovendo pesi o strutture meno importanti, con l'obiettivo di ottenere modelli più compatti e veloci a inference time. Il magnitude pruning è il metodo più semplice: si rimuovono i pesi con valore assoluto $|w_{ij}|$ più piccolo, sotto la soglia che i pesi grandi sono quelli che contribuiscono di più all'output. Nonostante la semplicita, è spesso sorprendentemente efficace come punto di partenza.

Il **pruning strutturato** rimuove intere strutture (canali, teste di attenzione, layer) per ottenere

accelerazione reale sull'hardware (il pruning non-strutturato produce matrici sparse difficili da accelerare). La varianza importanza di un filtro/testa k:

$$\mathcal{I}_k = \left|\frac{\partial \mathcal{L}}{\partial s_k}\right| \cdot |s_k| \quad \text{(Taylor expansion del costo di rimozione)}$$

Si rimuovono i k con più bassa importanza — rimpiazzati da zero per forzare la sparsità o rimossi fisicamente.

Knowledge Distillation

La **knowledge distillation** (Hinton et al., 2015) è una tecnica concettualmente diversa dalla compressione diretta: invece di eliminare parti del modello, si trasferisce la "conoscenza" da un modello teacher grande a uno student piccolo. Il meccanismo chiave sono le *soft targets* — le distribuzioni di probabilità del teacher, non le classi hard — che contengono informazione sulla struttura delle similarità tra classi molto più ricca delle label one-hot:

$$\mathcal{L}_{\text{KD}} = (1-\alpha)\mathcal{L}_{\text{CE}}(y, \sigma(z_S)) + \alpha T^2 \cdot \mathcal{L}_{\text{CE}}(\sigma(z_T/T), \sigma(z_S/T))$$

dove z_T e z_S sono i logit di teacher e student, T è la temperatura di distillazione (tipicamente $T = 3$–10) e α bilancia le due componenti. La temperatura alta "ammorbidisce" le probabilità del teacher, amplificando le probabilità delle classi non-corrette: queste piccole probabilità codificano la struttura delle similarità tra classi (es. "9" è più simile a "7" che a "1") — informazione preziosa che va ben oltre il semplice segnale della classe corretta.

Margini, Kernel Methods e Generalizzazione per Reti Profonde

133b.1 SVM e Bound da Margini

Per classificatori con margine γ (SVM), il bound sulla generalizzazione non dipende dalla dimensionalità ma solo dal margine:

Bound da Margini (Vapnik, 1998)

Per classificatori lineari con $\|\mathbf{w}\| \leq B$ su dati con $\|\mathbf{x}\| \leq R$, con probabilità $\geq 1-\delta$:

$$R_\gamma(h) \leq \hat{R}^m_\gamma(h) + O\left(\sqrt{\frac{(BR/\gamma)^2}{m} + \frac{\log(1/\delta)}{m}}\right)$$

dove R_γ è il risk con margine γ e $\hat{R}^m_\gamma$ è l'empirical margin loss.

La quantità $(BR/\gamma)^2$ è il **margin-normalized complexity** — non dipende da d. Un SVM con grande margine generalizza bene anche in alta dimensione.

133b.2 Kernel Methods e Reproducing Kernel Hilbert Spaces

Un kernel $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ è una funzione simmetrica definita positiva — definisce un prodotto scalare in uno spazio di Hilbert (RKHS) $\mathcal{H}_k$: $k(\mathbf{x}, \mathbf{x}') = \langle \phi(\mathbf{x}), \phi(\mathbf{x}') \rangle_{\mathcal{H}_k}$.

Mercer's theorem: k è definita positiva se e solo se $\sum_{i,j} c_i c_j k(x_i, x_j) \geq 0$ per ogni $\{c_i\}$ e $\{x_i\}$. Kernel comuni:

Kernel	Formula	Caratteristiche
Lineare	$k(\mathbf{x}, \mathbf{x}') = \mathbf{x}^T\mathbf{x}'$	Corrisponde a reti con attivazione lineare
Polinomiale	$k(\mathbf{x}, \mathbf{x}') = (\mathbf{x}^T\mathbf{x}' + c)^d$	Feature polinomiali implicite
RBF/Gaussiano	$k(\mathbf{x}, \mathbf{x}') = e^{-\|\mathbf{x}-\mathbf{x}'\|^2/2\sigma^2}$	Smooth, infinito-dimensionale
Neural Network (NNGP)	Ricorsione in profondità	Corrisponde a reti infinite (§91 Vol. II)

Il **Representer Theorem**: per qualsiasi problema di ottimizzazione regolarizzata in $\mathcal{H}_k$, la soluzione ottimale f^* è nella span dei kernel valutati sui training points: $f^*(\mathbf{x}) = \sum_{i=1}^m \alpha_i k(\mathbf{x}, \mathbf{x}_i)$. Questo riduce ottimizzazione in spazio infinito-dimensionale a un problema in $\mathbb{R}^m$.

133b.3 Uniform Convergence e le sue Limitazioni

I bound di generalizzazione classici (VC, Rademacher) si basano sulla uniform convergence: con molti campioni, l'errore empirico converge uniformemente all'errore reale per tutte le funzioni in $\mathcal{H}$ simultaneamente. Questo è sufficiente per la generalizzazione.

Nagarajan e Kolter (2019) dimostrano che i bound di uniform convergence sono necessariamente **vacui** per reti profonde overparametrizzate — non spiegano la generalizzazione osservata. I bound implicano $\Omega(N)$ errore per $N \gg m$, mentre in pratica le reti generalizzano con errore quasi zero.

Questo apre la domanda fondamentale: *perché le reti overparametrizzate generalizzano?* Le

risposte correnti:

- **Implicit regularization di SGD:** SGD trova la soluzione a norma minima (§91 Vol. II, NTK)
- **Flat minima:** i minima trovati da SGD sono "piatti" (bassa sharpness) — correlano con buona generalizzazione
- **Compressibility:** le reti addestrate su dati reali hanno bassa complessità di descrizione (MDL)
- **Benign overfitting:** per distribuzioni con code a bassa varianza (§133)

Flash Attention, Ring Attention e Comunicazione Collettiva

137b.1 Flash Attention: IO-Aware Self-Attention

Flash Attention (Dao et al., 2022) riformula il calcolo dell'attenzione per essere IO-aware — riduce il numero di letture/scritture dalla HBM (High Bandwidth Memory della GPU) invece di ridurre i FLOPs totali.

L'attenzione standard richiede materializzare la matrice $\mathbf{A} \in \mathbb{R}^{N \times N}$ in HBM: costo IO $O(N^2)$. Flash Attention usa il tiling: divide $\mathbf{Q}, \mathbf{K}, \mathbf{V}$ in blocchi che stanno nella SRAM (cache on-chip) e calcola l'attenzione in blocchi, accumulando il risultato:

$$\mathbf{O}_i = \text{softmax}(\mathbf{Q}_i \mathbf{K}^T / \sqrt{d}) \mathbf{V} = \text{softmax}\left(\frac{1}{\sqrt{d}} [\mathbf{Q}_i \mathbf{K}_1^T, \ldots, \mathbf{Q}_i \mathbf{K}_J^T] \right) [\mathbf{V}_1; \ldots; \mathbf{V}_J]$$

Il trucco: il softmax si può calcolare in modo online (algoritmo di Milakov, 2018) mantenendo solo le statistiche running max e normalization factor, senza materializzare la matrice completa:

$$m_j = \max(m_{j-1}, \max_k(\mathbf{Q}_i \mathbf{K}_j^T / \sqrt{d})), \quad l_j = e^{m_{j-1} - m_j} l_{j-1} + \sum_k e^{s_{jk} - m_j}$$

$$\mathbf{O}_j = \text{diag}(l_j)^{-1} \left(e^{m_{j-1} - m_j} \text{diag}(l_{j-1}) \mathbf{O}_{j-1} + \sum_k e^{s_{jk} - m_j} \mathbf{V}_{jk} \right)$$

Costo IO: $O(N^2 d/M)$ dove M è la dimensione della SRAM. Per $d = 64$ e $M = 100$KB, risparmio $\sim 64\times$ nel trasferimento dati. Flash Attention 2 e 3 ottimizzano ulteriormente per le architetture moderne (parallelismo per warp, interleaving di compute e memory transfers).

137b.2 Ring Attention: Contesti Ultra-Lunghi

Ring Attention (Liu et al., 2023) parallelizza l'attenzione su K device in un anello circolare per sequenze ultra-lunghe. Ogni device k mantiene il blocco $\mathbf{Q}_k$ e un blocco corrente di $\mathbf{K}_k, \mathbf{V}_k$. In ogni step:

Algoritmo: Ring Attention — Step per device k

per $t = 0, \ldots, K-1$:

Calcola blocco attenzione $\mathbf{Q}_k$ con $\mathbf{K}_{(k-t) \bmod K}, \mathbf{V}_{(k-t) \bmod K}$

Accumula risultato con online softmax

Invia $\mathbf{K}_{(k-t) \bmod K}, \mathbf{V}_{(k-t) \bmod K}$ al device $(k+1) \bmod K$

Ricevi blocco dal device $(k-1) \bmod K$

La comunicazione e il calcolo si overlappano — costo comunicazione $O(Nd/K)$ per device. Ring Attention permette sequenze di lunghezza $N = 1M+$ su cluster di GPU, scalando quasi linearmente con il numero di device.

137b.3 Collective Communication Primitives

Le operazioni collettive sono primitive di comunicazione su cui si basano tutti gli algoritmi distribuiti:

Operazione	Input/Output	Costo (ring)	Uso
AllReduce (sum)	Ogni device: tensore $X_k \to$ tensore $\sum_k X_k$	$2(K-1)\|X\|/K$	Gradient aggrega
AllGather	Ogni device: chunk $X_k \to$ tensore $[X_1; \cdots; X_K]$	$(K-1)\|X\|/K$	ZeRO-3, Ring At
ReduceScatter	Ogni device: tensore $X_k \to$ chunk $\sum_k X_k[i]$	$(K-1)\|X\|/K$	ZeRO-2/3, Mega
Broadcast	Root: tensore $X \to$ tutti ricevono X	$(K-1)\|X\|/K$	Parameter sync
AllToAll	Device k: chunk $X_{k,j}$ per $j \to$ riceve $X_{j,k}$	$(K-1)\|X\|/K$	Expert routing (

AllReduce = ReduceScatter + AllGather — due operazioni in sequenza, ognuna con costo $(K-1)|X|/K$, per un totale di $2(K-1)|X|/K$. Il ring-AllReduce è ottimale — ogni link fisico trasferisce esattamente $2|X|/K$ byte indipendentemente da K.

137b.4 Speculative Decoding: Inferenza Accelerata

Speculative Decoding (Chen et al., 2023) accelera la generazione autoregressiva usando un modello draft M_q (piccolo e veloce) e un modello target M_p (grande e accurato). Il draft genera γ token in sequenza; il target verifica tutti in parallelo (singolo forward pass):

$$\text{Accetta } x_{t+k} \text{ se } \mathcal{U}[0,1] \leq \frac{p(x_{t+k}|x_{<t+k})}{q(x_{t+k}|x_{<t+k})}$$

Se accettato, si mantiene il token. Se rifiutato, si campiona da una distribuzione corretta: $p'(x) \propto \max(0, p(x) - q(x))$. Il processo garantisce che la distribuzione dei token generati sia esattamente p (il target) — nessuna degradazione di qualità. Lo speedup: se il draft ha alta acceptance rate α, lo speedup medio è $\frac{\gamma+1}{1+\gamma(1-\alpha)}$. Per $\alpha = 0.8$ e $\gamma = 4$: speedup $\sim 3\times$.

DETR: Detection Transformer e Hungarian Matching

138.1 Architettura DETR

DETR (Carion et al., 2020) porta una rivoluzione concettuale nel rilevamento di oggetti: invece di ancorare predizioni a posizioni predefinite (anchor boxes) e applicare Non-Maximum Suppression per eliminarle, riformula il rilevamento come un **problema di predizione di insiemi**. Il pipeline è elegante nella sua semplicità:

1. CNN backbone estrae feature map $\mathbf{F} \in \mathbb{R}^{H' \times W' \times C}$
2. Flatten + positional encoding 2D $\to$ sequenza di $H'W'$ token
3. Encoder Transformer (self-attention globale sulla feature map)

4. $N = 100$ **object queries** $\mathbf{q}_1, \ldots, \mathbf{q}_N$ (learned embeddings)

5. Decoder Transformer: ogni query attende alle feature encoded (cross-attention) e alle altre query (self-attention)

6. FFN per ogni query: predice (c_i, b_i) — classe e bounding box

Le $N = 100$ predizioni vengono prodotte in parallelo (non autoregressivamente) da N query learnable che attendono alle feature dell'immagine. La maggior parte delle predizioni sarà "no object" ($\emptyset$), poiché immagini tipiche contengono molti meno di 100 oggetti. Questo schema elimina completamente anchor boxes, NMS e tutti i componenti ingegneristici domain-specific che caratterizzavano i detector precedenti.

138.2 Hungarian Matching Loss

Il problema centrale è come assegnare le N predizioni ai M ground-truth boxes ($M \leq N$) in modo che la loss sia minimizzabile con gradient descent. Un assegnamento fisso (es. predizione i deve predire il box i-esimo) non funziona perché non è invariante alla permutazione degli oggetti. DETR cerca invece l'assegnamento biunivoco ottimale $\hat{\sigma}$ che minimizza il costo totale:

$$\hat{\sigma} = \arg\min_{\sigma \in \mathfrak{S}_N} \sum_{i=1}^{N} \mathcal{C}_{\text{match}}(y_i, \hat{y}_{\sigma(i)})$$

dove il costo di abbinamento tra ground-truth $y_i = (c_i, b_i)$ e predizione $\hat{y}_j = (\hat{p}_j(c_i), \hat{b}_j)$:

$$\mathcal{C}_{\text{match}}(y_i, \hat{y}_j) = -\mathbf{1}[c_i \neq \emptyset]\hat{p}_j(c_i) + \mathbf{1}[c_i \neq \emptyset]\mathcal{L}_{\text{box}}(b_i, \hat{b}_j)$$

L'assegnamento ottimale si trova con l'**algoritmo ungherese** in $O(N^3)$ — trattabile per $N = 100$.

Algoritmo Ungherese (Hungarian Algorithm)

Data una matrice di costo $\mathbf{C} \in \mathbb{R}^{N \times N}$, l'algoritmo ungherese trova l'assegnamento ottimale in $O(N^3)$:

Algoritmo: Algoritmo Ungherese (versione Kuhn-Munkres)

1. Sottrai il minimo di ogni riga da ogni elemento della riga

2. Sottrai il minimo di ogni colonna da ogni elemento della colonna

3. Copri tutti gli zeri con il minimo numero di righe/colonne

4. Se righe di copertura $= N$: assegnamento ottimale trovato

5. Altrimenti: sottrai il minimo degli elementi non coperti, aggiungilo agli elementi doppiamente coperti, ripeti da 3

Loss di predizione per oggetti abbinati

$$\mathcal{L}_{\text{DETR}} = \sum_{i=1}^{N} \Big[-\log \hat{p}_{\hat{\sigma}(i)}(c_i) + \mathbf{1}[c_i \neq \emptyset] \mathcal{L}_{\text{box}}(b_i, \hat{b}_{\hat{\sigma}(i)}) \Big]$$

$$\mathcal{L}_{\text{box}}(b, \hat{b}) = \lambda_{\text{iou}} \mathcal{L}_{\text{GIoU}}(b, \hat{b}) + \lambda_{L1} \| b - \hat{b} \|_1$$

La combinazione GIoU + L1 normalizza la loss rispetto alla scala della box: GIoU cattura l'overlap relativo mentre L1 penalizza le deviazioni assolute di centro e dimensioni. Le predizioni abbinate a $\emptyset$ contribuiscono solo con una cross-entropy downweighted (fattore 0.1) sulla classe, prevenendo che il modello si specializzi nel predire "no object" su tutte le query. DETR converge lentamente (500 epoche contro 12 di Faster R-CNN) ma ha performance comparabili su COCO e si generalizza naturalmente a task di segmentazione e rilevamento panoptico.

SAM: Segment Anything Model

139.1 Architettura Promptable

SAM (Segment Anything Model, Kirillov et al., 2023) è il primo sistema di segmentazione universale zero-shot promtable: dato qualsiasi tipo di prompt (punto, box, testo, maschera), segmenta l'oggetto corrispondente in qualsiasi immagine. Il sistema è addestrato su SA-1B, un dataset di 1 miliardo di maschere annotate semiautomaticamente, e si generalizza a domini radicalmente diversi dal training. L'architettura è composta da tre componenti:

Image encoder: ViT-H/16 pre-addestrato con MAE, produce embedding $\mathbf{F} \in \mathbb{R}^{64\times64\times256}$ (downsampling 16×). L'encoder è il componente più costoso — eseguito una volta per immagine.

Prompt encoder: converte vari tipi di prompt in embedding:

- Punti (x, y, label) → positional encoding 2D + apprenduto offset
- Box (x_1, y_1, x_2, y_2) → embedding dei due corner
- Maschera → convoluzione 2D + embedding
- Testo → CLIP text encoder (opzionale)

Mask decoder: un Transformer leggero con due innovazioni. I **token output** (appresi) cross-atendono agli embedding dell'immagine e del prompt; poi, per ogni token, una MLP predice la maschera corrispondente. Il decoder usa **Two-Way Attention**: attenzione token→immagine e immagine→token, permettendo il flusso di informazione in entrambe le direzioni:

$$\mathbf{q} \leftarrow \text{Self-Attn}(\mathbf{q}) + \text{Cross-Attn}(\mathbf{q}, \mathbf{k})$$

$$\mathbf{k} \leftarrow \text{Cross-Attn}(\mathbf{k}, \mathbf{q})$$

SAM predice sempre 3 maschere candidate per gestire l'ambiguità intrinseca del prompt: un clic su un occhio potrebbe riferirsi all'occhio, alla faccia, o all'intera persona. Per ogni maschera viene predetto un confidence score; l'utente può selezionare la granularità desiderata. Questo design "ambiguity-aware" è uno dei contributi più originali di SAM rispetto ai detector precedenti che producevano una singola predizione deterministica.

SAM-2: Video Segmentation

SAM-2 (Ravi et al., 2024) estende SAM a video usando uno **streaming memory**: i frame precedenti segmentati vengono compressi in una "memoria" che informa la segmentazione dei frame successivi tramite cross-attention.

DeiT, BEiT e MAE-ViT: Avanzamenti nei Vision Transformer

140.1 DeiT: Data-Efficient Image Transformers

DeiT (Touvron et al., 2021) risolve un limite pratico fondamentale dei Vision Transformer: la necessità di dataset enormi come JFT-300M per il pre-training. DeiT dimostra che un ViT addestrato su ImageNet (1.2M immagini, non 300M) con la tecnica del **distillation token** raggiunge performance competitive con i migliori CNN. L'idea: aggiungere un token speciale di distillazione che deve predire le etichette di un teacher ResNet invece delle label one-hot standard:

$$\mathcal{L}_{\text{DeiT}} = (1-\lambda)\mathcal{L}_{\text{CE}}(y_{\text{true}}, z_{\text{cls}}) + \lambda\mathcal{L}_{\text{KD}}(y_{\text{teacher}}, z_{\text{dist}})$$

Il distillation token impara rappresentazioni complementari a quelle del CLS token: il teacher ResNet ha un inductive bias convoluzionale che codifica informazioni locali e di texture in modo diverso dal ViT, e il distillation token trasferisce questo bias strutturale al Transformer. A inference time, le predizioni del CLS token e del distillation token vengono combinate, beneficiando di entrambe le prospettive.

BEiT: Masked Image Modeling con DALL-E tokenizer

BEiT (Bao et al., 2022) introduce il masked image modeling ispirandosi a BERT nel NLP: così come BERT predice token testuali mascherati, BEiT predice **visual tokens** — rappresentazioni discrete delle patch ottenute da un tokenizer visivo (dVAE pre-addestrato con DALL-E). Invece di ricostruire i pixel grezzi (troppo facile, conduce a rappresentazioni di bassa qualità semantica), il modello classifica la patch mascherata tra un vocabolario di 8192 visual tokens:

$$\mathcal{L}_{\text{BEiT}} = -\sum_{i \in \mathcal{M}} \log P_\theta(v_i | \mathbf{x}_{\setminus \mathcal{M}})$$

dove v_i è il visual token della patch mascherata i e $\mathbf{x}_{\setminus\mathcal{M}}$ sono le patch visibili. La predizione di token discreti è un task di classificazione più facile da ottimizzare rispetto alla regressione sui pixel. BEiT→BEiT-3 scala il pre-training a testo e immagini congiuntamente.

Temporal Fusion Transformer

141.1 Architettura e Gating Gerarchico

Il forecasting in contesti reali raramente presenta la struttura pulita dei benchmark accademici. Consideriamo il forecasting delle vendite di un rivenditore: si hanno dati storici di vendita (osservati solo nel passato), informazioni sul calendario e promozioni future (note in anticipo), attributi statici dei negozi (invarianti nel tempo). Un Transformer standard tratta queste variabili in modo identico, perdendo la struttura temporale e causale che è al cuore del problema. Il Temporal Fusion Transformer (TFT, Lim, Arik, Loeff e Pfister, Google Cloud AI 2021) è progettato esplicitamente per il **forecasting multi-orizzonte** con variabili eterogenee di questo tipo — alcune note nel futuro (festivi, calendario), alcune osservate solo nel passato (vendite storiche), alcune statiche per entità (ID del negozio, categoria prodotto). Invece di trattare tutte le variabili uniformemente come farebbe un Transformer standard, TFT usa meccanismi specializzati per ciascun tipo. I tre componenti principali sono:

Gated Residual Network (GRN): unità di base del TFT con gating per la selezione contestuale:

$$\mathrm{GRN}(\mathbf{a}, \mathbf{c}) = \mathrm{LayerNorm}(\mathbf{a} + \mathrm{GLU}(\eta_1))$$

$$\eta_1 = \mathbf{W}_1 \eta_2 + \mathbf{b}_1, \quad \eta_2 = \mathrm{ELU}(\mathbf{W}_2 \mathbf{a} + \mathbf{W}_3 \mathbf{c} + \mathbf{b}_2)$$

$$\mathrm{GLU}(\mathbf{x}) = \sigma(\mathbf{W}_4 \mathbf{x} + \mathbf{b}_4) \odot (\mathbf{W}_5 \mathbf{x} + \mathbf{b}_5)$$

Variable Selection Network: seleziona le variabili rilevanti usando softmax sui pesi appresi. Per m variabili con embedding $\mathbf{e}^{(j)}$:

$$\mathbf{v}_\chi = \mathrm{Softmax}\big(\mathrm{GRN}\big([\mathbf{e}^{(1)}; \cdots; \mathbf{e}^{(m)}], \mathbf{c}_s\big)\big), \quad \tilde{\mathbf{e}} = \sum_j v_\chi^{(j)} \tilde{\mathbf{e}}^{(j)}$$

Multi-head attention con interpretabilità: TFT usa self-attention con condivisione dei pesi delle key per interpretabilità — ogni testa usa le stesse key/values, permettendo di visualizzare su quali timestep passati si concentra ogni predizione futura.

N-BEATS: Basis Expansion Analysis for Time Series

142.1 Decomposizione per Basis Expansion

N-BEATS (Oreshkin et al., 2020) porta un'idea insolita nel forecasting: una rete puramente fully-connected, senza convoluzioni né meccanismi di attenzione, basata sul principio che ogni serie temporale si decompone in una combinazione di basi (trend, stagionalità, residui). Ogni blocco nella rete produce due output: un **backcast** (ricostruzione dell'input) e un **forecast** (predizione futura), entrambi espressi come combinazioni lineari di vettori di base:

$$[\hat{\boldsymbol{\theta}}_f, \hat{\boldsymbol{\theta}}_b] = h_\ell(\mathbf{x}) \quad \text{(FC layers: lookback window} \to \text{coefficienti)}$$

$$\hat{\mathbf{y}}_\ell = \mathbf{V}_f \hat{\boldsymbol{\theta}}_f \quad \text{(forecast: proiezione su base forward)}$$

$$\hat{\mathbf{x}}_\ell = \mathbf{V}_b \hat{\boldsymbol{\theta}}_b \quad \text{(backcast: proiezione su base backward)}$$

$$\mathbf{x}_{\ell+1} = \mathbf{x}_\ell - \hat{\mathbf{x}}_\ell \quad \text{(residual connection)}$$

Le basi $\mathbf{V}_f$ e $\mathbf{V}_b$ sono:

- **Generic block:** learnable (FC layer che produce la base)
- **Trend block:** base polinomiale $\mathbf{V}_f = [1, t, t^2, \ldots, t^p]$, p piccolo (3–4)
- **Seasonality block:** base di Fourier $\mathbf{V}_f = [\cos(2\pi k t/H), \sin(2\pi k t/H)]_{k=1}^{\lfloor H/2 \rfloor}$

La struttura a stack con backcast è ispirata al principio della scomposizione progressiva: il primo stack apprende a rimuovere il trend dalla serie, il secondo la stagionalità, il terzo cattura i residui. I backcast vengono sottratti dall'input prima di passare allo stack successivo — un meccanismo di skip connection inverso che forza ogni stack a specializzarsi su un componente diverso della serie. Questo approccio è completamente interpretabile: si può visualizzare il contributo di trend e stagionalità separatamente.

PatchTST: Patch Tokenization per Serie Temporali

143.1 Tokenizzazione a Patch e Channel Independence

I primi Transformer applicati alle serie temporali tokenizzavano ogni singolo punto come un token — una scelta che ignorava la quasi-continuità del segnale. PatchTST (Nie, Ma, Li, Shah e Kossaifi, ICLR 2023) adatta l'architettura Vision Transformer alle serie temporali con due innovazioni ortogonali che risolvono i problemi strutturali dei Transformer temporali precedenti, ispirandosi a come ViT ha rivoluzionato la visione: la patch tokenization e il channel-independent modeling.

Patch tokenization: invece di trattare ogni timestep come un token (context length $\to$ numero di token, quadratico in attenzione), si aggregano P timestep consecutivi in una patch. Per una serie di lunghezza L con patch di dimensione P e stride S:

$$N = \left\lfloor \frac{L - P}{S} \right\rfloor + 2 \quad \text{numero di patch}$$

Tipicamente $P = 16$, $S = 8$: con $L = 336$ si ottengono 42 token invece di 336 — riduzione $8\times$ del costo di attenzione. Ogni patch è proiettata linearmente: $\mathbf{u}_i = \mathbf{W}_p \mathbf{x}_{(i-1)S+1:iS} + \mathbf{b}_p$.

Channel independence: per serie multivariate, ogni canale (variabile) viene processato indipendentemente dallo stesso Transformer. Non c'è cross-canale mixing nell'attenzione — il modello impara la dinamica temporale di ogni variabile separatamente. Questo riduce l'overfitting su dataset con molte variabili e permette il trasferimento tra dataset con numero diverso di canali.

Masked Pretraining per PatchTST

PatchTST con pre-training mascherato (analoga a MAE ma per serie temporali): si mascherano casualmente il 40% delle patch e si impara a ricostruirle. Il pre-training su grandi dataset di serie temporali migliora significativamente il fine-tuning su dataset piccoli — la conoscenza temporale è trasferibile tra domini.

iTransformer e Mamba per Serie Temporali

144.1 iTransformer: Inverted Attention

iTransformer (Liu et al., 2024) parte da un'osservazione critica: i Transformer temporali applicano l'attenzione tra i timestep, ma questo è computazionalmente costoso e spesso meno

informativo delle correlazioni tra variabili. iTransformer inverte i ruoli: ogni *variabile* (canale) diventa un token, e la sua serie temporale intera diventa il suo embedding. L'attenzione avviene tra le C variabili, catturando correlazioni cross-variate a costo $O(C^2)$ invece che $O(L^2)$:

$$\mathbf{h}_c = \mathbf{W}_{\text{embed}}\mathbf{x}_{c,1:L} + \mathbf{b}_{\text{embed}} \quad \forall c = 1, \ldots, C$$

L'attenzione avviene tra i C canali — cattura le correlazioni variate invece delle dipendenze temporali. Le dipendenze temporali sono modellate dalle operazioni FFN all'interno di ogni canale.

Risultato sorprendente: su molti benchmark di long-term forecasting, iTransformer supera tutti i Transformer temporali precedenti — compreso PatchTST — su serie multivariate con molte variabili correlate. Questo conferma l'ipotesi che le correlazioni tra variabili siano spesso più informative per il forecasting delle dipendenze temporali intra-variabile, almeno nell'orizzonte lungo.

Mamba per Serie Temporali

Mamba (Sezione 58 del Vol. II) si adatta naturalmente alle serie temporali grazie alla sua struttura ricorrente con memoria selettiva. L'input-selective SSM sceglie quali parti del contesto passato mantenere nella memoria — adatto per serie con dipendenze a lungo raggio selettive (eventi rari ma importanti).

Per serie multivariate Mamba offre due strategie complementari: (1) processare ogni canale indipendentemente con Mamba condividendo i pesi (approccio channel-independent come PatchTST, efficiente e spesso competitivo); (2) interleave Mamba layers per le dipendenze temporali e Attention layers per le correlazioni cross-canale (approccio ibrido che cattura entrambe le strutture a costo moderato). La scelta dipende dalla natura delle correlazioni nel dataset specifico.

Confronto degli approcci per Long-Term Forecasting

Modello	Complessità	Punto di forza	Punto debole
Transformer (Autoformer)	$O(L \log L)$	Attenzione a lungo raggio	Cattura correlazioni s
PatchTST	$O(N^2)$, $N \ll L$	Patch = contesto locale	Nessuna interazione t
iTransformer	$O(C^2)$	Correlazioni tra variabili	Contesto temporale f
Mamba-TS	$O(L)$	Memoria selettiva, efficiente	Training più instabile
N-BEATS	$O(L)$	Interpretabile, nessuna attenzione	Solo univariato
TFT	$O(L^2)$	Variabili eterogenee, interpretabile	Architettura comples

---

Contrastive Language-Image Pretraining: CLIP e Varianti

140b.1 CLIP: Fondamenti e Derivazione della Loss

CLIP (Radford et al., 2021) addestra un encoder di testo T_ϕ e un encoder di immagini I_ψ su coppie (immagine, testo) usando una loss contrastiva simmetrica. Dato un batch di N coppie $\{(\mathbf{x}_i, t_i)\}_{i=1}^N$:

$$\mathbf{I}_i = l_2\text{-norm}(I_\psi(\mathbf{x}_i)), \quad \mathbf{T}_i = l_2\text{-norm}(T_\phi(t_i))$$

$$\text{logit}_{ij} = \mathbf{I}_i^T \mathbf{T}_j \cdot e^\tau \quad (\tau\text{: parametro di temperatura apprendo})$$

$$\mathcal{L}_{\text{CLIP}} = \frac{1}{2N}\left[\sum_i \mathcal{L}_{\text{CE}}(\text{logit}_{i,:}, i) + \sum_j \mathcal{L}_{\text{CE}}(\text{logit}_{:,j}, j)\right]$$

La prima somma ottimizza l'alignment immagine→testo (per ogni immagine, il suo testo deve essere il più simile); la seconda testo→immagine. Addestrando su 400M di coppie web-scraped, CLIP impara rappresentazioni open-vocabulary che permettono zero-shot classification su qualsiasi dataset.

Zero-Shot con CLIP

Per classificare un'immagine in K classi senza fine-tuning: si codificano i template testuali "a photo of a class" per ogni classe, si calcolano le similarità con l'embedding dell'immagine, e si

predice la classe con similarità massima:

$$\hat{y} = \arg\max_k \mathbf{I}^T \mathbf{T}_k$$

CLIP raggiunge $\sim 76\%$ su ImageNet zero-shot — comparabile a ResNet-50 con supervisione. La chiave: i template testuali catturano la semantica delle classi, e l'encoder di immagini ha imparato un allineamento accurato con lo spazio testuale.

140b.2 GroundingDINO e Open-Set Detection

GroundingDINO (Liu et al., 2023) estende DINO al rilevamento open-set — trovare oggetti descritti in linguaggio naturale senza classi predefinite. L'architettura fonde feature di testo (BERT) e immagine (Swin Transformer) tramite un modulo di feature fusion bidirezionale:

$$\text{ImgFeature}' = \text{ImgFeature} + \text{CrossAttn}(\text{ImgFeature}, \text{TextFeature})$$

$$\text{TextFeature}' = \text{TextFeature} + \text{CrossAttn}(\text{TextFeature}, \text{ImgFeature})$$

La matching loss alterna tra il matching bipartito (come DETR) e la text-guided supervision — le query dell'object detector sono inizializzate dai token di testo invece che da learned embeddings.

140b.3 Diffusion per Computer Vision: ControlNet

ControlNet (Zhang et al., 2023) aggiunge controllo spaziale a un modello di diffusione pre-addestrato (Stable Diffusion) senza modificare i pesi originali. Crea una copia trainable del blocco encoder e la condiziona sul segnale di controllo (edge, depth, pose):

$$\mathbf{y} = \mathcal{F}(\mathbf{x}; \boldsymbol{\Theta}) + \mathcal{Z}(\mathcal{F}(\mathbf{x} + \mathcal{Z}(\mathbf{c}; \boldsymbol{\Theta}_1); \boldsymbol{\Theta}_c); \boldsymbol{\Theta}_2)$$

dove $\mathcal{Z}$ sono layer con zero-initialization (zero convolution: peso e bias inizializzati a zero). All'inizio del training, il contributo del ControlNet è zero — il modello originale è preservato. Man mano che i pesi crescono, il ControlNet aggiunge informazione spaziale incrementalmente.

Anomaly Detection e Forecasting con Incertezza

144b.1 Anomaly Detection nelle Serie Temporali

Il rilevamento di anomalie nelle serie temporali può essere formulato come stima della densità: un punto $\mathbf{x}_t$ è anomalo se $p(\mathbf{x}_t|\mathbf{x}_{<t})$ è bassa. Metodi principali:

LSTM-VAE: un LSTM-encoder e un LSTM-decoder in framework VAE producono un punteggio di anomalia basato sulla reconstruction error e sulla KL divergence:

$$\text{score}_t = \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x}_t)}[-\log p_\theta(\mathbf{x}_t|\mathbf{z})] + D_{\text{KL}}(q_\phi(\mathbf{z}|\mathbf{x}_t)\|p(\mathbf{z}))$$

Anomaly Transformer (Xu et al., 2022): usa un **association discrepancy** tra la prior-association (gaussiana locale) e la series-association (attenzione appresa). Le anomalie hanno alta discrepanza — la serie non si comporta come previsto dall'associazione locale. L'obiettivo di training usa un minimax game tra le due associazioni per massimizzare la discrepanza in training e minimizzarla per i pattern normali.

144b.2 Conformal Prediction per Serie Temporali

La conformal prediction (§84, Vol. II) si applica al forecasting per produrre intervalli di predizione con garanzie di copertura marginale. Per serie temporali, il problema è la dipendenza temporale — le assunzioni exchangeability del conformal classico non tengono.

Conformal PID (Angelopoulos et al., 2024) adatta il livello di copertura dinamicamente per serie temporali con distribuzione non stazionaria, usando un controllore PID sull'errore di copertura:

$$q_{t+1} = q_t + k_p e_t + k_i \sum_{s \le t} e_s + k_d (e_t - e_{t-1})$$

dove $e_t = \alpha - \mathbf{1}[y_t \notin C_t]$ è l'errore di copertura al passo t e $C_t = [\hat{y}_t - q_t, \hat{y}_t + q_t]$ è l'intervallo. Il controllore PID garantisce copertura media $1-\alpha$ anche per distribuzioni che cambiano nel tempo.

144b.3 Foundation Models per Serie Temporali

I recenti foundation models per time series cercano un unico modello pre-addestrato su migliaia di dataset eterogenei, capace di zero-shot forecasting su nuovi dataset.

TimesFM (Das et al., 2024, Google): decoder-only Transformer pre-addestrato su $\sim 10^{12}$ timestep da Google Trends, Wikipedia traffic, e dataset pubblici. Usa patch tokenization (come PatchTST) con patch di dimensione variabile e un meccanismo di output per la previsione multi-orizzonte. La generalizzazione zero-shot su 11 benchmark di forecasting supera molti metodi specializzati per-dataset.

Moirai (Salesforce, 2024): usa patch di dimensioni multiple in parallelo (multi-patch tokenization) per catturare la stagionalità a diverse frequenze. L'attenzione è applicata tra tutte le patch di tutte le frequenze — il modello impara pattern multi-scala in modo unificato.

144b.4 Tabella Riassuntiva: Quando Usare Quale Approccio

Scenario	Approccio consigliato	Motivo
Dataset piccolo (< 1000 osservazioni)	N-BEATS, ETS classico	Poca capacità per Transfor
Molte variabili correlate	TFT, iTransformer	Cattura correlazioni cross-
Contesto molto lungo (> 1000)	PatchTST, Mamba	Efficienza con sequenze lur
Dataset nuovo senza dati storici	TimesFM, Moirai (zero-shot)	Foundation model per tran
Interpretabilità richiesta	N-BEATS (interpretable), TFT	Componenti esplicite
Incertezza rigorosa	Conformal + qualsiasi modello	Garanzie di copertura mar
Anomaly detection real-time	Anomaly Transformer, LSTM-VAE	Reconstruction-based scori

VAE Gerarchici, GAN con Gradient Penalty: Derivazioni Complete

150.1 Hierarchical VAE: LVAE e NVAE

I VAE gerarchici (HVAE) modellano distribuzioni più espressive usando latenti a più livelli $\mathbf{z}_L, \ldots, \mathbf{z}_1$ con prior autoregressivo:

$$p(\mathbf{x}, \mathbf{z}_{1:L}) = p(\mathbf{x}|\mathbf{z}_1) \prod_{\ell=1}^{L} p(\mathbf{z}_\ell | \mathbf{z}_{\ell+1:L})$$

L'ELBO si fattorizza su tutti i livelli:

$$\mathcal{L}_{\text{HVAE}} = \mathbb{E}_q[\log p(\mathbf{x}|\mathbf{z}_1)] - \sum_{\ell=1}^{L} D_{\text{KL}}(q(\mathbf{z}_\ell|\mathbf{z}_{<\ell}, \mathbf{x}) \| p(\mathbf{z}_\ell|\mathbf{z}_{\ell+1:L}))$$

Il problema del KL vanishing: i layer superiori tendono a "saltare" la prior (KL $\to 0$) — il modello ignora i livelli alti della gerarchia. **NVAE** (Vahdat e Kautz, 2020) risolve con: (1) residual normal distributions — le posteriori sono espresse come perturbazioni della prior; (2) batch norm calibration; (3) spectral regularization sui layer delle convoluzioni.

LADDER-VAE usa una struttura top-down dove le posteriori condizionano sequenzialmente dall'alto verso il basso:

$$q(\mathbf{z}_\ell|\mathbf{z}_{\ell+1}, \mathbf{x}) = \mathcal{N}(\boldsymbol{\mu}_q(\mathbf{z}_{\ell+1}, \mathbf{x}), \boldsymbol{\sigma}_q^2(\mathbf{z}_{\ell+1}, \mathbf{x}))$$

$$\boldsymbol{\mu}_q = \frac{\boldsymbol{\sigma}_p^2 \boldsymbol{\mu}_x + \boldsymbol{\sigma}_x^2 \boldsymbol{\mu}_p}{\boldsymbol{\sigma}_p^2 + \boldsymbol{\sigma}_x^2}, \quad \boldsymbol{\sigma}_q^2 = \frac{\boldsymbol{\sigma}_p^2 \boldsymbol{\sigma}_x^2}{\boldsymbol{\sigma}_p^2 + \boldsymbol{\sigma}_x^2}$$

La fusione gaussiana ponderata combina le informazioni dal prior $(\boldsymbol{\mu}_p, \boldsymbol{\sigma}_p^2)$ (top-down) e dall'encoder bottom-up $(\boldsymbol{\mu}_x, \boldsymbol{\sigma}_x^2)$ — è la posterior esatta per due gaussiane indipendenti.

150.2 Derivazione Completa WGAN: Dualità di Kantorovich-Rubinstein

La Wasserstein-1 distance tra P e Q:

$$W_1(P, Q) = \inf_{\gamma \in \Pi(P,Q)} \mathbb{E}_{(\mathbf{x},\mathbf{y})\sim\gamma}[\|\mathbf{x} - \mathbf{y}\|]$$

dove $\Pi(P, Q)$ è l'insieme di tutti i joint distributions con marginali P e Q. Per la dualità di Kantorovich-Rubinstein:

Teorema (Kantorovich-Rubinstein, 1958)

$$W_1(P, Q) = \sup_{\|f\|_L \leq 1} \mathbb{E}_{\mathbf{x}\sim P}[f(\mathbf{x})] - \mathbb{E}_{\mathbf{y}\sim Q}[f(\mathbf{y})]$$

dove il sup è su tutte le funzioni 1-Lipschitz $f : \mathcal{X} \to \mathbb{R}$, cioè $|f(\mathbf{x}) - f(\mathbf{y})| \leq \|\mathbf{x} - \mathbf{y}\|$.

Prova sketch: per ogni coupling $\gamma \in \Pi(P, Q)$ e funzione f 1-Lipschitz:

$\mathbb{E}_P[f] - \mathbb{E}_Q[f] = \mathbb{E}_\gamma[f(\mathbf{x}) - f(\mathbf{y})] \leq \mathbb{E}_\gamma[\|\mathbf{x} - \mathbf{y}\|]$ Per la Lipschitz condition: $f(\mathbf{x}) - f(\mathbf{y}) \leq \|\mathbf{x} - \mathbf{y}\|$

Prendendo l'inf sui coupling: $\sup_f(\mathbb{E}_P[f] - \mathbb{E}_Q[f]) \leq W_1(P, Q)$ Il bound vale per qualsiasi f 1-Lipschitz e qualsiasi coupling

Per l'uguaglianza: il coupling ottimale γ^* e la funzione f^* (potenziale di Kantorovich) saturano il bound Esistenza garantita per spazi polacchi compatti — teorema di Prokhorov

150.3 Normalizing Flows: Change of Variables Generalizzato

Il change of variables formula per una trasformazione invertibile $f : \mathbb{R}^d \to \mathbb{R}^d$ (difeomorfismo):

$$p_{\mathbf{x}}(\mathbf{x}) = p_{\mathbf{z}}(f(\mathbf{x})) \cdot |\det \mathbf{J}_f(\mathbf{x})|$$

dove $\mathbf{J}_f = \partial \mathbf{z} / \partial \mathbf{x}$ è il Jacobiano di f. In log:

$$\log p_{\mathbf{x}}(\mathbf{x}) = \log p_{\mathbf{z}}(f(\mathbf{x})) + \log |\det \mathbf{J}_f(\mathbf{x})|$$

Per una composizione di trasformazioni $f = f_K \circ \cdots \circ f_1$ (con $\mathbf{z}_0 = \mathbf{x}$ e $\mathbf{z}_K = \mathbf{z}$):

$$\log p(\mathbf{x}) = \log p(\mathbf{z}_K) + \sum_{k=1}^{K} \log |\det \mathbf{J}_{f_k}(\mathbf{z}_{k-1})|$$

La logica del "flow": si parte da una distribuzione semplice $p(\mathbf{z}_K)$ (es. gaussiana standard) e si trasforma con K invertible maps sempre più espressive — la distribuzione risultante $p(\mathbf{x})$ può approssimare qualsiasi distribuzione continua.

Calcolo del Determinante Jacobiano per diverse architetture

Trasformazione	$\log \lvert \det \mathbf{J} \rvert$	**Costo**
Coupling additivo (NICE)	0	$O(1)$
Coupling affine (RealNVP)	$\sum_{i>k} s_i(x_{1:k})$	$O(d-k)$
Autoregressive (MAF)	$\sum_i \log \lvert \sigma_i(x_{<i}) \rvert$	$O(d)$
Invertible 1×1 Conv (Glow)	$H \cdot W \cdot \log \lvert \det \mathbf{W} \rvert$	$O(C^3)$
LU decomposition (Glow)	$H \cdot W \cdot \sum_i \log \lvert s_i \rvert$	$O(C)$
Residual Flow	Power series (Hutchinson)	$O(d)$ approx
FFJORD (Neural ODE)	$-\int_0^T \operatorname{tr}(\mathbf{J}_f)\, dt$	$O(d)$ Hutchinson

Equazioni di Bellman Avanzate: Average Reward, Semi-MDP e Options

151.1 Average Reward MDP

La formulazione discounted reward con $\gamma < 1$ è conveniente matematicamente — garantisce la convergenza della somma infinita e privilegia le ricompense immediate — ma introduce un'artificiosità concettuale in molti problemi reali: perché dovrebbe importare quando una ricompensa viene ricevuta in un task di controllo continuo senza fine naturale? In applicazioni come il controllo robotico, i giochi con partite molto lunghe, o la gestione di sistemi industriali, il discount factor distorce la policy ottimale verso comportamenti miopi che accumulano ricompense presto a scapito del comportamento a lungo termine. In questi scenari — task continui senza terminal state — il criterio più naturale è la ricompensa media per timestep, e il fattore di sconto $\gamma < 1$ non è appropriato. Il **average reward criterion** ottimizza il reward medio per timestep:

$$\rho^\pi = \lim_{T\to\infty} \frac{1}{T} \mathbb{E}_\pi \left[\sum_{t=0}^{T-1} R_t \right]$$

L'equazione di Bellman del average reward (**Bellman relative value equation**):

$$\rho^* + V^*(s) = \max_a \left[R(s,a) + \sum_{s'} P(s'|s,a) V^*(s') \right]$$

dove $V^*(s)$ è la funzione di valore *relativo* (definita a meno di una costante). La soluzione è unica a meno di un offset additivo su V^* — si fissa $V^*(s_0) = 0$ per uno stato di riferimento s_0.

151.2 Options Framework: RL su Più Scale Temporali

Le **Options** (Sutton et al., 1999) estendono l'MDP con azioni temporalmente estese. Un'option $\omega = (I_\omega, \pi_\omega, \beta_\omega)$ consiste di:

- $I_\omega \subseteq \mathcal{S}$: insieme degli stati in cui l'option può essere iniziata
- $\pi_\omega : \mathcal{S} \times \mathcal{A} \to [0,1]$: intra-option policy
- $\beta_\omega : \mathcal{S} \to [0,1]$: termination function (probabilità di terminare in ogni stato)

L'**option-value function** e la Bellman equation per le options:

$$Q_\Omega(s,\omega) = \sum_a \pi_\omega(a|s) \sum_{s'} P(s'|s,a)[R(s,a) + \gamma\,[(1-\beta_\omega(s'))Q_\Omega(s',\omega) + \beta_\omega(s')V_\Omega(s')]]$$

Il primo termine nella parentesi quadra: l'option continua con prob. $1-\beta_\omega$ — si usa ancora $Q_\Omega(s',\omega)$. Il secondo termine: l'option termina con prob. β_ω — si sceglie la migliore option da s' tramite $V_\Omega(s') = \max_{\omega'} Q_\Omega(s',\omega')$.

Identità Matematiche per RL, Modelli Generativi e Teoria Statistica

152.1 Identità Fondamentali del Reinforcement Learning

Relazioni fondamentali tra value functions	
$V^\pi(s) = \sum_a \pi(a\|s) Q^\pi(s,a)$	State-value da action-value
$Q^\pi(s,a) = R(s,a) + \gamma \sum_{s'} P(s'\|s,a) V^\pi(s')$	Action-value da state-value
$A^\pi(s,a) = Q^\pi(s,a) - V^\pi(s)$	Advantage function
$V^\pi(s) = \mathbb{E}_{a\sim\pi}[A^\pi(s,a)] = 0$	Advantage ha media zero
$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}[\nabla \log \pi_\theta(a\|s) \cdot Q^\pi(s,a)]$	Policy Gradient Theorem
$\hat{A}_t^{\text{GAE}} = \sum_{l=0}^{\infty} (\gamma\lambda)^l \delta_{t+l}$	GAE advantage
$r_t(\theta) = \pi_\theta(a_t\|s_t) / \pi_{\theta_{\text{old}}}(a_t\|s_t)$	Importance ratio PPO
$\delta_t = R_{t+1} + \gamma V(S_{t+1}) - V(S_t)$	TD error
$V^\pi_{\text{soft}}(s) = \mathbb{E}_{a\sim\pi}[Q^\pi_{\text{soft}}(s,a) + \alpha H(\pi)]$	Soft value (SAC)
$\pi^*_{\text{soft}}(a\|s) \propto \exp(Q^*_{\text{soft}}(s,a)/\alpha)$	Policy ottimale SAC

152.2 Identità per Modelli Generativi

VAE e modelli latenti	
$\log p(\mathbf{x}) \geq \mathcal{L}_{\text{ELBO}} = \mathbb{E}_{q_\phi}[\log p_\theta(\mathbf{x}\|\mathbf{z})] - D_{\text{KL}}(q_\phi\|p)$	ELBO — lower bound
$\log p(\mathbf{x}) - \mathcal{L}_{\text{ELBO}} = D_{\text{KL}}(q_\phi(\mathbf{z}\|\mathbf{x})\|p(\mathbf{z}\|\mathbf{x}))$	Gap = posterior KL
$D_{\text{KL}}(\mathcal{N}(\boldsymbol{\mu}, \text{diag}(\boldsymbol{\sigma}^2))\|\mathcal{N}(\mathbf{0}, \mathbf{I})) = -\frac{1}{2}\sum_j(1 + \log\sigma_j^2 - \mu_j^2 - \sigma_j^2)$	KL gaussiana-standard in forma chiusa
Diffusion e score matching	
$q(\mathbf{x}_t\|\mathbf{x}_0) = \mathcal{N}(\sqrt{\bar{\alpha}_t}\mathbf{x}_0, (1-\bar{\alpha}_t)\mathbf{I})$	Forward marginal DDPM
$\tilde{\boldsymbol{\mu}}_t = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\varepsilon}\right)$	Reverse mean DDPM
$\mathbf{s}_\theta(\mathbf{x}_t, t) = -\boldsymbol{\varepsilon}_\theta(\mathbf{x}_t, t)/\sqrt{1-\bar{\alpha}_t}$	Score = noise normalizzato
$W_1(P, Q) = \sup_{\|f\|_L \leq 1}[\mathbb{E}_P[f] - \mathbb{E}_Q[f]]$	Dualità Kantorovich-Rubinstein
Normalizing Flows	
$\log p(\mathbf{x}) = \log p_Z(f(\mathbf{x})) + \log\lvert\det \mathbf{J}_f(\mathbf{x})\rvert$	Change of variables
$\log\lvert\det \mathbf{J}\rvert = 0$ per coupling additivo	NICE — Jacobiano unitario
$\log\lvert\det \mathbf{J}\rvert = \sum_{i>k} s_i(x_{1:k})$ per coupling affine	RealNVP — diagonale a blocchi
$\log\lvert\det \mathbf{W}\rvert \to \sum_i \log\lvert s_i\rvert$ con LU decomposition	Glow invertible 1×1 conv

152.3 Identità per Teoria Statistica

Generalizzazione e complessità	
$R(h) \leq \hat{R}_m(h) + \sqrt{\frac{2d_{\text{VC}}\log(em/d_{\text{VC}})}{m}} + O(\sqrt{\log(1/\delta)/m})$	Bound VC (finita, caso finito)
$R(h) \leq \hat{R}_m(h) + 2\mathfrak{R}_m(\mathcal{H}) + \sqrt{\frac{\ln(1/\delta)}{2m}}$	Bound Rademacher
$\mathfrak{R}_m(\mathcal{H}_B) \leq \frac{BR}{\sqrt{m}}$ per funzioni lineari	Rademacher lineare
$I(\mathbf{x}; \mathbf{x}') = H(\mathbf{x}') - H(\mathbf{x}'\|\mathbf{x})$	Mutual information augmentation view
$R_T \geq \sum_{k:\Delta_k>0} \frac{\log T}{\Delta_k}$ per MAB (Lai-Robbins)	Lower bound regret bandit
$R_T^{\text{UCB}} \leq O\left(\sqrt{KT\log T}\right)$	Upper bound UCB1

Debugging per RL, Modelli Generativi e Self-Supervised

153.1 Checklist RL: Diagnostica dei Problemi Comuni

Sintomo	Causa probabile	Rimedio
Reward cresce poi collassa	Policy diverge (update troppo grande)	Diminuire LR, usa
Reward non cresce mai	Reward sparso / exploration insufficiente	Reward shaping, c
Instabilità nelle Q-values	Mancanza target network o replay buffer	Aggiungere target
DQN overestima sistematicamente	Max bias	Passare a Double
Policy ripete la stessa azione	Mode collapse / entropia troppo bassa	Aumentare coeffici
Convergenza lenta in continuous control	Reward scale sbagliata	Normalizzare rewa
Policy peggiora durante RLHF	Reward hacking / KL troppo alto	Aumentare β nella
SAC non converge	Temperatura α mal calibrata	Abilitare automati

153.2 Checklist GAN: Diagnostica

Sintomo	Causa probabile	Rimedio
Immagini generate tutte uguali	Mode collapse completo	Mini-batch discr
Loss D $\to 0$, loss G $\to \infty$	Discriminatore troppo forte	Rallentare aggio
Artefatti a scacchiera	Deconvoluzioni (frequency artifacts)	Usare upsample
FID non migliora dopo molte epoche	Training bloccato in equilibrio subottimale	Restart training
NaN nei gradienti	Log(0) nella loss o gradient explosion	Aggiungere eps,
VAE genera immagini sfocate	Penalità KL troppo alta (β grande)	KL annealing, d
VAE posterior collassa (KL$\to$0 per dim)	Decoder troppo potente	Ridurre capacità

153.3 Checklist Self-Supervised: Diagnostica

Sintomo	Causa probabile	Rimedio
Representations costanti (collapse)	Mancanza asimmetria o negativi	Aggiungere stop-g
Loss scende ma linear probe scarsa	Augmentation elimina features semantiche	Ridurre intensità
SimCLR non scala con batch size	Batch troppo piccolo, pochi negativi	Aumentare batch
MAE ricostruisce ma features scarse	Masking ratio troppo bassa	Aumentare maski
DINO collapse improvviso	Centering non funziona / temperatura errata	Verificare EMA c
Few-shot accuracy bassa con MAML	Inner LR troppo alto/basso	Grid search inner

153.4 Iperparametri Consigliati per Volume III

Modello	Iperparametro chiave	Valore di partenza	Range tipico
PPO	Clipping ε	0.2	0.1–0.3
GAE λ	0.95	0.9–0.99	
Entropy coeff c_2	0.01	0.001–0.05	
SAC	Temperatura α (auto)	Auto	$-\lvert\mathcal{A}\rvert$ target entropy
Replay buffer size	10^6	10^5–10^7	
Soft update τ	0.005	0.001–0.01	
DDPM	T (timestep)	1000	500–2000
Schedule	cosine	linear, cosine, sigmoid	
Model dim / UNet	128	64–256	
SimCLR	Temperatura τ	0.07	0.05–0.2
Projection dim	128	64–512	
Batch size	4096	256–8192	
MAML	Inner LR α	0.01	0.001–0.1
Inner steps	5	1–10	
WGAN-GP	λ (gradient penalty)	10	1–100
n_{critic}	5	3–10	

GAN: Instabilità nell'Addestramento — Analisi Dinamica

154.1 Training GAN come Sistema Dinamico

Il training simultaneo di G e D con gradient descent non converge in generale. Consideriamo il caso semplice: D parametrizzato da $\phi \in \mathbb{R}^p$ e G da $\theta \in \mathbb{R}^q$. Il gradient field del gioco è:

$$\dot{\phi} = \nabla_\phi V(G_\theta, D_\phi), \quad \dot{\theta} = -\nabla_\theta V(G_\theta, D_\phi)$$

Linearizzando attorno all'equilibrio (ϕ^*, θ^*), il Jacobiano del sistema è:

$$\mathbf{J}_{\text{game}} = \begin{pmatrix} \mathbf{A} & \mathbf{B} \\ -\mathbf{C} & -\mathbf{D} \end{pmatrix}$$

dove $\mathbf{A} = \nabla^2_\phi V$, $\mathbf{D} = \nabla^2_\theta V$, $\mathbf{B} = \nabla_{\phi,\theta} V$, $\mathbf{C} = \nabla_{\theta,\phi} V$. Il sistema è stabile se tutti gli autovalori di $\mathbf{J}_{\text{game}}$ hanno parte reale negativa.

Il problema: per un gioco convesso-concavo V, gli autovalori di $\mathbf{J}_{\text{game}}$ sono **puramente immaginari** — il sistema orbita attorno all'equilibrio senza convergere. Gradient Descent Ascent (GDA) non converge nemmeno per i giochi bilineari più semplici $V(\phi, \theta) = \phi^T \mathbf{C} \theta$.

Extragradient method (Korpelevich, 1976) risolve questo problema con uno step di lookahead:

$$\phi^{t+1/2} = \phi^t + \eta \nabla_\phi V(\phi^t, \theta^t), \quad \theta^{t+1/2} = \theta^t - \eta \nabla_\theta V(\phi^t, \theta^t)$$

$$\phi^{t+1} = \phi^t + \eta \nabla_\phi V(\phi^{t+1/2}, \theta^{t+1/2}), \quad \theta^{t+1} = \theta^t - \eta \nabla_\theta V(\phi^{t+1/2}, \theta^{t+1/2})$$

L'extragradient converge per giochi convesso-concavi Lipschitz con tasso $O(1/T)$ — è stabile dove GDA oscilla.

154.2 Collapse Analysis nei Modelli Contrastive

Perché i metodi contrastive senza negativi (BYOL, SimSiam) non collassano a soluzioni costanti? Analisi formale per SimSiam (Chen e He, 2021).

SimSiam usa stop-gradient: $\mathcal{L} = -\frac{1}{2}\text{sim}(p_1, z_2^{\text{stop}}) - \frac{1}{2}\text{sim}(p_2, z_1^{\text{stop}})$. Il gradiente dell'obiettivo è:

$\nabla_\theta \mathcal{L} = -\frac{1}{2}[\nabla_\theta \text{sim}(p_1, z_2) + \nabla_\theta \text{sim}(p_2, z_1)]\big|_{z_1^{\text{stop}}, z_2^{\text{stop}}}$ Il stop-gradient elimina la dipendenza da $\nabla_\theta z_i$

Alla soluzione costante $p_1 = p_2 = z_1 = z_2 = \mathbf{c}$: $\nabla_\theta \mathcal{L} = 0$ è soddisfatto L'equilibrio costante è un punto fisso dell'update rule

Ma il predictor q_θ è addestrato a predire z: se $z = \mathbf{c}$, q apprende $p = \mathbf{c}$ — ma allora la loss è $-\text{sim}(\mathbf{c}, \mathbf{c}) = -1$ (minima già) Il collapse è un minimum locale della loss — non è prevenuto dalla struttura della loss!

La risoluzione: il predictor q rimuove il "dead equilibrium" dinamicamente. Se z collassa, q smette di avere un gradiente utile — il predictor non può imparare la mappatura identità $z \mapsto z$ quando tutte le rappresentazioni sono uguali. In pratica, il BN (Batch Normalization) nell'encoder previene il collapse forzando $\text{Var}(z_i) > 0$ all'interno del batch.

PAC-Bayes e Bound Tight per Reti Neurali

155.1 PAC-Bayes Framework

Il I classici bound di generalizzazione — VC dimension, Rademacher complexity — si applicano al classificatore peggiore in una classe, producendo spesso bound troppo larghi per essere informativi. Il PAC-Bayes framework (McAllester 1999, ampliato da Catoni e Seeger) offre un'alternativa più flessibile e spesso più stretta: fornisce bound sulla generalizzazione per *distribuzioni* di classificatori, sfruttando l'informazione prior sulla struttura dei pesi.

Teorema PAC-Bayes (McAllester, 1999 — versione Catoni)

Con probabilità $\geq 1-\delta$ su $S \sim \mathcal{D}^m$, per qualsiasi posterior Q:

$$\mathbb{E}_{h\sim Q}[R(h)] \leq \mathbb{E}_{h\sim Q}[\hat{R}_m(h)] + \sqrt{\frac{D_{\mathrm{KL}}(Q\|P) + \ln(1/\delta)}{2m}}$$

Il bound dipende da $D_{\mathrm{KL}}(Q\|P)$ — quanto la posterior si è "spostata" rispetto al prior. Per prior gaussiano $P = \mathcal{N}(\mathbf{0}, \sigma^2\mathbf{I})$ e posterior gaussiano $Q = \mathcal{N}(\boldsymbol{\theta}, \sigma^2\mathbf{I})$:

$$D_{\mathrm{KL}}(Q\|P) = \frac{\|\boldsymbol{\theta}\|^2}{2\sigma^2}$$

Quindi: modelli con norma piccola $\|\boldsymbol{\theta}\|$ hanno bound migliori — questo formalizza perché la regolarizzazione L2 aiuta la generalizzazione. La connessione con SAM (§80, Vol. II e §133b): SAM ottimizza implicitamente per una posterior che rimane vicina al prior (flat region dell'ottimizzatore).

155.2 Implicit Regularization e NTK

Il NTK (§91, Vol. II) mostra che nelle reti infinite, il training con gradient flow converge alla soluzione di minimum-norm interpolation. Il bound di generalizzazione corrispondente:

$$R(\hat{f}_m) \leq O\left(\sqrt{\frac{\|\mathbf{y}\|^2_{(K_\infty)^{-1}}}{m}}\right)$$

dove $\|\mathbf{y}\|^2_{(K_\infty)^{-1}} = \mathbf{y}^T K_\infty^{-1}\mathbf{y}$ è la norma RKHS della funzione target nel kernel space del NTK. Questo bound è indipendente dal numero di parametri — dipende solo dalla complessità della funzione target nel kernel space.

Transformer per RL: Decision Transformer e Gato

156.1 Decision Transformer: RL come Sequence Modeling

Decision Transformer (Chen et al., 2021) riformula il RL come un problema di next-token prediction su sequenze di (return, state, action). Invece di ottimizzare la policy tramite backup di Bellman, si condiziona su un **return-to-go** desiderato $\hat{R}_t = \sum_{t'=t}^{T} r_{t'}$ e si predice l'azione corrispondente:

$$a_t = \text{Transformer}(\hat{R}_1, s_1, a_1, \hat{R}_2, s_2, a_2, \ldots, \hat{R}_t, s_t)$$

A inference time, si inserisce il return-to-go desiderato (es. il massimo osservato nel dataset) e il modello genera le azioni per raggiungerlo — behaviour cloning condizionato. Non c'è bootstrapping, nessuna funzione di valore, nessuna policy gradient — il Transformer impara a "imitare" le traiettorie del dataset con il return-to-go più alto.

Vantaggio: stabile su dataset offline, non soffre di distributional shift. Svantaggio: non migliora oltre il return massimo nel dataset — non impara a combinare sub-ottimali per creare nuove soluzioni ottime (stitching problem).

Gato: Generalist Agent

Gato (Reed et al., 2022) usa un singolo Transformer per oltre 600 task diversi — testo, immagini, Atari, robotica. Ogni modalità è tokenizzata separatamente (testo: BPE; immagini: ViT patch; azioni continue: μ-law encoding a 1024 valori; azioni discrete: dirette) e concatenata in una singola sequenza:

$$\underbrace{[P_{\text{task}}]}_{\text{context}} \underbrace{[I_1, \ldots, I_N]}_{\text{img tokens}} \underbrace{[s_1, a_1, s_2, a_2, \ldots]}_{\text{RL tokens}} \underbrace{[t_1, \ldots, t_M]}_{\text{text tokens}}$$

Un singolo next-token prediction objective è usato per tutti i task — il modello apprende a condizionare sul context per capire il task corrente. Gato dimostra positive transfer tra task: le performance sul singolo task migliorano con il multi-task training, suggerendo che la competenza in un task trasferisce ad altri.

UNet per Diffusione: Architettura Completa con Attention

157.1 UNet con Attention per DDPM

Il backbone standard per il noise-predicting network $\boldsymbol{\varepsilon}_\theta$ in DDPM è una UNet con residual blocks e self-attention alle risoluzioni intermedie. L'architettura:

$$\text{Input: } \mathbf{x}_t \in \mathbb{R}^{H \times W \times C}, \quad t \in \{1, \ldots, T\}$$

Il timestep t viene codificato con Sinusoidal Position Encoding (come nei Transformer) e aggiunto tramite adaptive layer normalization o aggiunto direttamente ai feature map:

$$\mathbf{e}_t = [\sin(\omega_k t), \cos(\omega_k t)]_{k=1}^{d/2}, \quad \omega_k = e^{-2k \log(T_{\max})/d}$$

$$\mathbf{h}'_l = \mathbf{h}_l + \mathbf{W}\mathbf{e}_t \quad \text{(time conditioning)}$$

Ogni **ResBlock** nella UNet:

$$\mathbf{h}' = \text{GroupNorm}(\mathbf{h}) \xrightarrow{\text{SiLU}} \text{Conv} \xrightarrow{+\mathbf{W}\mathbf{e}_t} \text{GroupNorm} \xrightarrow{\text{SiLU}} \text{Conv} \xrightarrow{+\text{ skip}} \mathbf{h}_{\text{out}}$$

Alle risoluzioni basse (es. 16×16 e 8×8), si inseriscono layer di **Multi-Head Self-Attention** — le feature spaziali si attendono globalmente. Questo permette al modello di catturare struttura globale dell'immagine pur operando con convoluzioni locali alle risoluzioni alte.

Cross-Attention per Condizionamento Testuale

Per modelli condizionati su testo (LDM, Stable Diffusion), ogni ResBlock include cross-attention tra le feature visive e i token testuali $\boldsymbol{\tau}_\theta(\mathbf{y})$:

$$\text{Attn}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d}}\right)\mathbf{V}$$

$$\mathbf{Q} = \mathbf{W}_Q \phi(\mathbf{x}_t), \quad \mathbf{K} = \mathbf{W}_K \boldsymbol{\tau}_\theta(\mathbf{y}), \quad \mathbf{V} = \mathbf{W}_V \boldsymbol{\tau}_\theta(\mathbf{y})$$

Le query provengono dalle feature visive, key e value dai token testuali — ogni pixel "chiede" alle parole del testo quale informazione semantica è rilevante per quella posizione spaziale.

ViT e Architetture Ibrida: Swin Transformer, MViT

158.1 Swin Transformer: Attenzione Locale con Window Shifting

Swin Transformer (Liu et al., 2021) risolve il problema dell'attenzione quadratica di ViT usando **window-based attention**: l'attenzione si calcola solo all'interno di finestre non-overlapping di $M \times M$ patch ($M = 7$). Il costo diventa $O(N \cdot M^2 d)$ invece di $O(N^2 d)$.

Il problema: nessuna comunicazione tra finestre diverse. La soluzione: **Shifted Window Attention** — in layer alternati, le finestre sono spostate di $(M/2, M/2)$:

$$\text{Layer } 2k\text{: finestre in } \{0, M, 2M, \ldots\} \times \{0, M, 2M, \ldots\}$$

$$\text{Layer } 2k+1\text{: finestre in } \{M/2, 3M/2, \ldots\} \times \{M/2, 3M/2, \ldots\}$$

Le due configurazioni alternate garantiscono che ogni patch comunichi con patch vicine da entrambe le griglie di finestre — connettività globale raggiunta in $O(\log N)$ layer. L'implementazione efficiente usa **cyclic shift** + masking per evitare di creare nuove finestre ai bordi.

Relative Position Bias

Swin aggiunge un bias di posizione relativa $B \in \mathbb{R}^{M^2 \times M^2}$ all'attention score:

$$\text{Attn}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d}} + B\right)\mathbf{V}$$

B è parametrizzato tramite una lookup table sui spostamenti relativi $(i - j) \in \{-(M-1), \ldots, M-1\}^2$ — i bias si condividono tra tutte le finestre. Il relative position bias è più

efficace dell'absolute positional encoding per il rilevamento perché cattura la struttura locale invariante alla posizione assoluta.

158.2 DINOv2: Foundation Model per Vision

DINOv2 (Oquab et al., 2023) scala DINO con: (1) dataset curato da 142M di immagini (LVD-142M, con de-duplication e quality filtering); (2) ViT-g/14 come backbone (1.1B parametri); (3) distillazione regolarizzata. Le features DINOv2 sono universali — un semplice linear probe ottiene $> 80\%$ su ImageNet e state-of-the-art su 12 benchmark senza fine-tuning specifico.

La coerenza semantica delle features DINOv2 è notevole: punti corrispondenti su oggetti semanticamente simili (due cani di razze diverse, due sedie di stili diversi) hanno alta similarità nel feature space — emerge senza supervisione esplicita di corrispondenza.

Contrastive Learning per Dati Tabulari e Multimodali

159.1 SCARF: Self-Supervised Contrastive Learning per Dati Tabulari

L'enorme successo del contrastive learning su immagini (SimCLR, MoCo) e testo (BERT) ha naturalmente sollevato la questione: questi approcci si generalizzano a dati tabulari, la forma più comune di dati nel mondo reale? La risposta non è ovvia: per le immagini esistono augmentazioni naturali e semanticamente significative (rotazione, ritaglio, jitter cromatico) che preservano il contenuto informativo. Per dati tabulari strutturati — dove ogni feature ha un significato specifico e la permutazione delle righe è priva di significato — le augmentazioni devono essere ripensate da zero. SCARF (Self-supervised Contrastive Learning for tabular data, Bahri et al. 2022) risolve questo problema con un'augmentation che sfrutta la struttura statistica del dataset: genera viste augmentate di righe tabulari corrompendo casualmente un sottoinsieme di feature e sostituendole con valori campionati dalla loro distribuzione marginale empirica, mantenendo le correlazioni intra-riga nelle feature non corrotte:

$$\tilde{x}_j = \begin{cases} x_j & \text{con prob. } 1 - p_{\text{corr}} \\ v_j \sim \hat{P}(x_j) & \text{con prob. } p_{\text{corr}} \end{cases}$$

La coppia positiva $(\mathbf{x}, \tilde{\mathbf{x}})$ è la riga originale e la sua versione corrotta. La loss contrastiva NT-Xent viene applicata sulle rappresentazioni — il modello impara a riconoscere che le feature corrotte appartengono allo stesso record originale.

Perché funziona: la corruzione marginale distrugge le correlazioni tra feature ma preserva

l'identità del record. Il modello impara rappresentazioni che catturano la struttura correlazionale dell'intera riga — più informative delle feature individuali.

159.2 ALIGN e Florence: Vision-Language Foundation Models

ALIGN (Jia et al., 2021) scala CLIP a 1.8B di coppie noisy image-text — dimostrando che la scala dei dati compensa la rumorosità delle annotations. Florence (Yuan et al., 2021) aggiunge gerarchia semantica con task specifici in fine-tuning (coarse-grained classification, fine-grained, detection, VQA) su una singola rappresentazione pre-addestrata.

Il pattern comune dei vision-language models: un large-scale contrastive pre-training su dati web produce un feature space allineato, che viene poi adattato a task specifici con piccoli dataset etichettati. Il transfer è efficiente: pochi shot su nuovi task, forte zero-shot su task non visti.

Teoria dell'Informazione Applicata al Machine Learning

160.1 Minimum Description Length e Bayesian Model Selection

La teoria dell'informazione e il machine learning condividono una domanda fondamentale: come scegliere il modello giusto tra quelli possibili? La regolarizzazione empirica (penalizzare la complessità del modello) ha un fondamento teorico profondo nella teoria di Kolmogorov e Shannon, che Shannon stesso aveva intuito: un modello migliore è un modello che comprime meglio i dati, perché comprendere significa individuare le regolarità che permettono una rappresentazione più concisa. Il principio MDL (Minimum Description Length, Rissanen 1978) formalizza questa intuizione: il modello migliore è quello che minimizza la lunghezza totale della descrizione del modello più la descrizione dei dati dato il modello. Il MDL seleziona quindi il modello che comprime meglio i dati: $M^* = \arg\min_M L(M) + L(\mathcal{D}|M)$. Per modelli probabilistici, il codice ottimale ha lunghezza $-\log p(\mathcal{D}|M)$ — la log-likelihood negativa. Con prior $P(M)$:

$$\text{MDL}(M, \mathcal{D}) = -\log P(M) - \log P(\mathcal{D}|M) = -\log P(M|\mathcal{D}) + \text{const}$$

MDL = MAP inference. La connessione con l'ELBO: minimizzare la loss VAE $-\mathcal{L}_{\text{ELBO}} = -\mathbb{E}_q[\log p(\mathbf{x}|\mathbf{z})] + D_{\text{KL}}(q\|p)$ è equivalente a minimizzare la lunghezza di descrizione dell'encoder $D_{\text{KL}}(q\|p)$ (bits per codificare $\mathbf{z}$) più la lunghezza di descrizione del decoder $-\log p(\mathbf{x}|\mathbf{z})$ (bits per codificare $\mathbf{x}$ dato $\mathbf{z}$).

160.2 Mutual Information Neural Estimation (MINE)

MINE (Belghazi et al., 2018) stima la mutual information $I(X;Z)$ tramite la dualità di Donsker-Varadhan:

$$I(X;Z) = \sup_{T:\Omega\to\mathbb{R}} \mathbb{E}_{p(x,z)}[T(x,z)] - \log \mathbb{E}_{p(x)p(z)}[e^{T(x,z)}]$$

dove il sup è su tutte le funzioni misurabili T. Si parametrizza T_θ come una rete neurale e si ottimizza θ con gradient ascent. MINE fornisce uno stimatore consistent e asintoticamente unbiased di $I(X;Z)$ — utile per diagnosi in VAE (quanto informazione effettivamente passa attraverso il latente?) e in contrastive learning.

Mixture of Experts: Routing, Capacità e Load Balancing

161.1 Struttura di un MoE Layer

Un **Mixture of Experts (MoE) layer** sostituisce l'FFN denso di un Transformer con E reti FFN ("expert") distinte. Solo k expert vengono attivati per ogni token — sparsità k/E. Il routing decide quali k expert ricevono ogni token:

$$\text{MoE}(\mathbf{x}) = \sum_{i=1}^{E} g_i(\mathbf{x}) \cdot \text{FFN}_i(\mathbf{x})$$

$$g_i(\mathbf{x}) = \begin{cases} \text{softmax}(\text{topk}(\mathbf{W}_r\mathbf{x}))_i & \text{se } i \in \text{Top-}k(\mathbf{W}_r\mathbf{x}) \\ 0 & \text{altrimenti} \end{cases}$$

Con $E = 64$ expert e $k = 2$, ogni token attiva $2/64 = 3.1\%$ dei parametri FFN — il costo di compute è quasi identico a un modello denso con $1/32$ degli expert, ma la *capacità* (numero di parametri) è $32\times$ più grande.

Il problema del Load Balancing

Senza regolarizzazione, il router collassa: pochi expert ricevono quasi tutti i token (popularity bias), il resto non viene mai addestrato. Il problema è analogo al mode collapse nei GAN. La **auxiliary load balancing loss** di GShard (Lepikhin et al., 2021) penalizza le distribuzioni sbilanciate:

$$\mathcal{L}_{\text{aux}} = \alpha E \sum_{i=1}^{E} f_i \cdot P_i$$

$$f_i = \frac{1}{T} \sum_{t=1}^{T} \mathbf{1}[\text{token } t \to \text{expert } i], \qquad P_i = \frac{1}{T} \sum_{t=1}^{T} g_i(\mathbf{x}_t)$$

dove f_i è la frazione di token assegnata all'expert i e P_i è la probabilità media di routing verso i. La loss $\mathcal{L}_{\text{aux}} \propto \sum_i f_i P_i$ è minimizzata quando tutti gli expert ricevono la stessa frazione $1/E$ di token — un distribution uniforme.

161.2 Switch Transformer: Top-1 Routing

Switch Transformer (Fedus et al., 2021) usa routing $k = 1$ (ogni token va a esattamente 1 expert) — massima sparsità e minor overhead. La **expert capacity** limita il numero di token per expert per evitare overflow:

$$\text{Capacity} = C \cdot \frac{T}{E} \cdot \frac{1}{k}$$

dove $C > 1$ è il capacity factor (tipicamente 1.25). Se un expert riceve più token del suo capacity, i token in eccesso vengono **dropped** (passano direttamente alla residual connection). Il capacity factor bilancia efficienza (basso C) e qualità (alto C).

Confronto Switch vs dense a parità di FLOPs

Switch-C (1.6T parametri, 2048 expert) converge 7× più velocemente di T5-XXL (11B parametri) su C4 pre-training — la stessa loss viene raggiunta con 7× meno steps. Il vantaggio viene principalmente dalla maggiore capacità del modello, non dall'efficienza computazionale per token.

161.3 Mixtral e Expert Specialization

Mixtral 8×7B (Jiang et al., 2024) usa $E = 8$ expert con $k = 2$ per layer FFN, per un totale di 46.7B parametri ma solo 12.9B parametri attivi per token. L'analisi della specializzazione: gli expert non si specializzano semanticamente in modo netto (non ci sono "expert del testo legale" e "expert del codice") — invece si specializzano su strutture sintattiche e stilistiche locali. La specializzazione è emergente e distribuita.

La formula di routing di Mixtral è identica a Switch ma con $k = 2$: il token viene processato dai due FFN con i router weights più alti, e l'output è la loro combinazione pesata con softmax normalizzato sui top-2 weights.

Linear Attention e Efficient Transformers: Survey Matematico

162.1 Il Kernel Trick per l'Attenzione

L'attenzione softmax $\text{Attn}(\mathbf{Q},\mathbf{K},\mathbf{V})_i = \frac{\sum_j k(\mathbf{q}_i,\mathbf{k}_j)\mathbf{v}_j}{\sum_j k(\mathbf{q}_i,\mathbf{k}_j)}$ con $k(\mathbf{q},\mathbf{k}) = \exp(\mathbf{q}^T\mathbf{k}/\sqrt{d})$ ha costo $O(N^2)$. La **linear attention** approssima con un kernel fattorizzato $k(\mathbf{q},\mathbf{k}) = \phi(\mathbf{q})^T\phi(\mathbf{k})$:

$$\text{LinearAttn}(\mathbf{Q},\mathbf{K},\mathbf{V})_i = \frac{\phi(\mathbf{q}_i)^T\Big(\sum_j \phi(\mathbf{k}_j)\mathbf{v}_j^T\Big)}{\phi(\mathbf{q}_i)^T\sum_j \phi(\mathbf{k}_j)}$$

Il termine $\mathbf{S} = \sum_j \phi(\mathbf{k}_j)\mathbf{v}_j^T \in \mathbb{R}^{d_\phi \times d_v}$ può essere accumulato in $O(Nd_\phi d_v)$ — lineare in N. La query $\mathbf{q}_i$ poi moltiplica $\mathbf{S}$ in $O(d_\phi d_v)$.

Scelte di ϕ

Metodo	$\phi(\mathbf{x})$	d_ϕ	**Qu**
Performer (FAVOR+)	Feature random della softmax kernel: $e^{-\|\mathbf{x}\|^2/2}[\sin(\boldsymbol{\omega}^T\mathbf{x}), \cos(\boldsymbol{\omega}^T\mathbf{x})]$	$r \ll d$	Bu
Linear Transformer	$\text{elu}(\mathbf{x}) + 1$	d	Ac
cosFormer	$[\cos(\pi it/2T), \sin(\pi it/2T)]$	$2d$	Co
RWKV (WKV)	Ricorrenza pesata esponenziale	d	Ot

162.2 Sparse Attention: Longformer e BigBird

Invece di approssimare il kernel, la **sparse attention** limita quali coppie (i,j) partecipano all'attenzione. Il pattern di sparsità:

$$A_{ij} = \begin{cases} \exp(\mathbf{q}_i^T\mathbf{k}_j/\sqrt{d}) & (i,j) \in \mathcal{S} \\ 0 & \text{altrimenti} \end{cases}$$

Longformer usa tre tipi di attenzione: (1) **local window** — ogni token attende a $w/2$ token adiacenti in ogni direzione ($O(Nw)$); (2) **global token** — pochi token speciali (es. [CLS])

attendono a tutti; (3) **dilated sliding window** — finestre con gap per contesti più grandi. Il costo complessivo: $O(N(w+g))$ dove g è il numero di global token.

BigBird (§82, Vol. II) aggiunge **random attention** per garantire la Turing-completezza — senza di essa, la window attention non può implementare operazioni che richiedono dipendenze globali arbitrarie.

162.3 State Space Models vs Attention: Analisi Comparativa

Mamba (§58, Vol. II) usa state space models (SSM) invece di attention. La connessione con l'attenzione lineare:

$$\text{SSM output: } y_t = \sum_{s=1}^{t} h(t,s)\, x_s \quad \text{(matrice di mixing causale)}$$

$$\text{Attenzione causale: } y_t = \sum_{s=1}^{t} \text{softmax}(\mathbf{q}_t^T \mathbf{k}_s)\mathbf{v}_s$$

La differenza chiave: in SSM, $h(t,s)$ è determinato dai parametri del modello applicati ai token separatamente; in attenzione, $h(t,s)$ dipende dalla similarità tra $\mathbf{q}_t$ e $\mathbf{k}_s$ — il mixing è *content-dependent*. Mamba usa un SSM con gate input-dependent ($\mathbf{B}, \mathbf{C}, \Delta$ in funzione dell'input) per approssimare il content-dependent mixing a costo lineare.

Caratteristica	**Attention**	**SSM (Mamba)**	**Linear Atte**
Costo training	$O(N^2 d)$	$O(Nd^2)$ parallelo	$O(Nd^2)$
Costo inferenza	$O(Nd)$ per token (con KV cache)	$O(d^2)$ per token (RNN)	$O(d^2)$ per to
Content-dependent	Sì (pieno)	Parziale (gates)	Sì (kernel ap
Recall su sequenze lunghe	Ottimo	Buono con selective SSM	Scarso (mem
Parallelismo training	Pieno	Pieno (parallel scan)	Pieno

GNN Message Passing: Formulazione Formale e Espressività

163.1 Message Passing Neural Networks (MPNN)

Il framework MPNN (Gilmer et al., 2017) unifica GCN, GraphSAGE, GAT e molte altre architetture sotto un'unica formulazione. Ogni layer MPNN esegue tre operazioni:

$$\mathbf{m}_v^{(t)} = \bigoplus_{u \in \mathcal{N}(v)} M_t(\mathbf{h}_v^{(t)}, \mathbf{h}_u^{(t)}, \mathbf{e}_{uv}) \qquad \text{(aggregazione messaggi)}$$

$$\mathbf{h}_v^{(t+1)} = U_t(\mathbf{h}_v^{(t)}, \mathbf{m}_v^{(t)}) \qquad \text{(aggiornamento nodo)}$$

dove M_t è la message function, $\bigoplus$ è un operatore di aggregazione permutation-invariant (sum, mean, max), U_t è la update function, e $\mathbf{e}_{uv}$ sono gli edge features.

Casi speciali:

Modello	$M_t(\mathbf{h}_v, \mathbf{h}_u, \mathbf{e})$	$\bigoplus$	$U_t(\mathbf{h}_v, \mathbf{m}_v)$
GCN	$\tilde{D}^{-1/2}\tilde{A}\tilde{D}^{-1/2}\mathbf{h}_u$	sum	$\sigma(\mathbf{W}\mathbf{m}_v)$
GraphSAGE	$\mathbf{h}_u$	mean/max/LSTM	$\sigma(\mathbf{W}[\mathbf{h}_v \Vert \mathbf{m}_v])$
GAT	$\alpha_{vu}\mathbf{W}\mathbf{h}_u$	sum	$\mathrm{ELU}(\sum \alpha_{vu}\mathbf{W}\mathbf{h}_u)$
MPNN (generale)	$\mathrm{MLP}(\mathbf{h}_v, \mathbf{h}_u, \mathbf{e}_{uv})$	sum	$\mathrm{GRU}(\mathbf{h}_v, \mathbf{m}_v)$

163.2 Espressività: Weisfeiler-Lehman Test

La capacità di una GNN di distinguere grafi non-isomorfi è legata al test di Weisfeiler-Lehman (WL) per l'isomorfismo dei grafi. WL-1 (il test standard) colora iterativamente i nodi aggregando i colori dei vicini:

$$c_v^{(t+1)} = \mathrm{HASH}\big(c_v^{(t)}, \{\{c_u^{(t)} : u \in \mathcal{N}(v)\}\}\big)$$

Teorema di Espressività delle GNN (Xu et al., 2019)

Qualsiasi GNN con aggregazione sum e update function iniettiva è al massimo potente quanto il WL-1 test — non può distinguere grafi che WL-1 non distingue. La GNN raggiunge questa potenza massima se e solo se: (1) usa aggregazione sum (non mean o max); (2) la update function è iniettiva. Le GNN con mean/max aggregation sono strettamente meno potenti.

Graph Isomorphism Network (GIN) implementa la GNN più potente possibile nel framework WL-1:

$$\mathbf{h}_v^{(k+1)} = \text{MLP}^{(k)}\left((1+\varepsilon^{(k)})\mathbf{h}_v^{(k)} + \sum_{u \in \mathcal{N}(v)} \mathbf{h}_u^{(k)}\right)$$

Il parametro ε (learnable o fisso a 0) pesa diversamente il contributo del nodo stesso rispetto ai vicini — garantisce l'iniettività dell'aggregazione per multiset con cardinalità diversa.

163.3 Oltre WL-1: Higher-Order GNN e Geometric GNN

Per distinguere grafi che WL-1 non distingue (es. due grafi 3-regolari non-isomorfi), servono test di ordine superiore. k-WL generalizza il test a k-tuple di nodi — esponenzialmente più espressivo ma computazionalmente costoso ($O(n^k)$ nodi).

Le **Geometric GNN** (EGNN, SchNet, DimeNet) sfruttano la struttura geometrica 3D delle molecole. La chiave: le funzioni apparse devono essere E(3)-equivarianti — trasformazioni rigide dello spazio (rotazioni, traslazioni, riflessioni) non cambiano le predizioni sulle proprietà molecolari.

$$f(\mathbf{R}\mathbf{x}_i + \mathbf{t}) = \mathbf{R} f(\mathbf{x}_i) \quad \forall \mathbf{R} \in \text{SO}(3), \mathbf{t} \in \mathbb{R}^3 \qquad \text{(equivarianza E(3))}$$

EGNN (§77, Vol. II) usa distanze interatomiche come input invarianti e coordinate relative per aggiornare le posizioni in modo equivariante. Le distanze $d_{ij} = \|\mathbf{x}_i - \mathbf{x}_j\|$ sono automaticamente invarianti a rotazioni e traslazioni.

Catastrophic Forgetting ed Elastic Weight Consolidation

164.1 Il Problema del Catastrophic Forgetting

Nel **continual learning** (o lifelong learning), un modello deve apprendere sequenzialmente una serie di task $\mathcal{T}_1, \mathcal{T}_2, \ldots$ senza dimenticare i precedenti. Il **catastrophic forgetting** (McCloskey e Cohen, 1989) è la tendenza delle reti neurali a perdere completamente le performance su $\mathcal{T}_1$ dopo il training su $\mathcal{T}_2$ — SGD sovrascrive i pesi ottimali per $\mathcal{T}_1$ senza vincoli.

Il **stability-plasticity dilemma**: alta plasticità (apprendimento veloce) porta a forgetting; alta stabilità (resistenza al cambiamento) impedisce l'apprendimento di nuovi task.

I tre scenari principali:

- **Task-Incremental:** il task ID è noto a test time — ogni task ha la sua testa di classificazione
- **Domain-Incremental:** il task ID non è noto, ma l'output space è fisso (stesse classi, diversa distribuzione)
- **Class-Incremental:** il task ID non è noto e nuove classi si aggiungono nel tempo — il più difficile

164.2 Elastic Weight Consolidation (EWC)

EWC (Kirkpatrick et al., 2017) protegge i pesi importanti per i task passati usando la Fisher Information Matrix come misura di importanza. L'idea: i pesi θ_i^* importanti per il task A (alta curvatura della loss in quella direzione) non devono cambiare molto durante il training su B.

La posterior bayesiana dopo i task A e B:

$\log p(\theta|\mathcal{D}_A, \mathcal{D}_B) = \log p(\mathcal{D}_B|\theta) + \log p(\theta|\mathcal{D}_A) - \log p(\mathcal{D}_B)$ Bayes su $\mathcal{D}_B$ usando $p(\theta|\mathcal{D}_A)$ come prior

$\log p(\theta|\mathcal{D}_A) \approx -\frac{1}{2}\sum_i F_i(\theta_i - \theta^*_{A,i})^2 + \text{const}$ Approssimazione di Laplace: la posterior è gaussiana con precisione data dalla Fisher $F_i = \mathbb{E}_{p(\mathbf{x}|\theta^*_A)}[(\partial_{\theta_i} \log p(\mathbf{x}|\theta))^2]$

Obiettivo EWC: $\mathcal{L} = \mathcal{L}_B(\theta) + \frac{\lambda}{2}\sum_i F_i(\theta_i - \theta^*_{A,i})^2$ Massimizzare la joint posterior = minimizzare la loss su B con penalità quadratica pesata da F_i

$$\mathcal{L}_{\text{EWC}} = \mathcal{L}_B(\theta) + \frac{\lambda}{2}\sum_i F_i(\theta_i - \theta^*_{A,i})^2$$

La Fisher diagonale F_i è la varianza del gradiente del log-likelihood rispetto al parametro i — alta varianza = il parametro è informativo per A = va protetto. Pesi con $F_i \approx 0$ sono "liberi" di cambiare per apprendere B.

Online EWC

Con molti task, EWC richiede di memorizzare una Fisher per ogni task precedente. Online EWC (Schwarz et al., 2018) mantiene una singola Fisher running average:

$$\tilde{F}^{(k)} = \gamma \tilde{F}^{(k-1)} + F^{(k)}$$

Questa compressione introduce un errore approssimativo — i task vecchi vengono dimenticati esponenzialmente con decay γ, permettendo una forma controllata di forgetting.

Replay Methods e Progressive Neural Networks

165.1 Experience Replay per Continual Learning

I **replay methods** mantengono un buffer $\mathcal{M}$ di esempi dai task passati e li mescolano con il training corrente. Il challenge principale: il buffer ha capacità limitata — quali esempi memorizzare?

GEM (Gradient Episodic Memory, Lopez-Paz e Ranzato, 2017) garantisce che ogni aggiornamento non aumenti la loss su nessun task passato. Dato il gradiente g del task corrente e i gradienti g_k del task k calcolati sul buffer, si proietta g sul cono fattibile:

$$\tilde{g} = \arg\min_{z} \|z - g\|^2 \quad \text{s.t. } \langle z, g_k \rangle \geq 0 \; \forall k$$

Il vincolo $\langle \tilde{g}, g_k \rangle \geq 0$ garantisce che l'aggiornamento $-\tilde{g}$ non aumenti la loss su k (il coseno tra il gradiente aggiornato e quello del task passato è non-negativo — si muovono nella stessa direzione). La soluzione si trova con dual quadratic programming in $O(K^2)$ dove K è il numero di task passati.

iCaRL (Rebuffi et al., 2017) per class-incremental learning mantiene un **exemplar set** — pochi rappresentanti per classe selezionati come i più vicini alla media della classe:

$$\mathbf{p}_y = \frac{1}{|\mathcal{X}_y|} \sum_{\mathbf{x} \in \mathcal{X}_y} \phi(\mathbf{x}) \qquad \text{(prototipo della classe } y\text{)}$$

La classificazione usa nearest-mean classifier con i prototipi — aggiornabili senza ri-training quando arrivano nuove classi.

165.2 Progressive Neural Networks e PackNet

Progressive Neural Networks (Rusu et al., 2016) non modificano mai i pesi dei task passati — aggiungono una nuova "colonna" di layer per ogni task, con **lateral connections** che leggono le feature dei task passati:

$$\mathbf{h}_i^{(t,k)} = f\left(\mathbf{W}^{(t,k)}\mathbf{h}^{(t,k-1)} + \sum_{j<t} \mathbf{U}_j^{(t,k)}\mathbf{h}^{(j,k-1)}\right)$$

dove $\mathbf{W}^{(t,k)}$ sono i pesi della colonna t al layer k e $\mathbf{U}_j^{(t,k)}$ sono le lateral connections dalla colonna j. Zero forgetting garantito — le colonne passate sono frozen. Il costo: la memoria cresce linearmente con il numero di task.

PackNet (Mallya e Lazebnik, 2018) usa iterative pruning per liberare capacità per nuovi task. Per ogni task, si allena il modello, si fa il pruning del $p\%$ dei pesi meno importanti (e si freezano quelli mantenuti), poi si riallena solo i pesi non-frozen per il task successivo. Il modello "impacchetta" $1/p^K$ task con buona performance con K round di pruning.

165.3 Dark Experience Replay e Riepilogo dei Metodi

DER++ (Buzzega et al., 2020) augmenta il replay standard con una knowledge distillation loss sulle logit memorizzate. Per ogni esempio nel buffer si memorizza anche il vettore di logit al momento dell'inserimento z_i:

$$\mathcal{L}_{\text{DER++}} = \mathcal{L}_{\text{CE}}(y, f(\mathbf{x})) + \alpha \mathbb{E}_{(\mathbf{x}_{\text{rep}}, z_{\text{rep}})\sim\mathcal{M}}\left[\|f(\mathbf{x}_{\text{rep}}) - z_{\text{rep}}\|^2\right] + \beta \mathcal{L}_{\text{CE}}(y_{\text{rep}}, f(\mathbf{x}_{\text{rep}}))$$

La seconda loss (distillazione sulle logit memorizzate) preserva non solo le predizioni corrette ma la struttura completa dello spazio delle probabilità — più informativa della sola CE con le etichette hard.

Famiglia	Metodo	Memoria aggiuntiva	Garantisce zero forgetti
Regularization	EWC	$O(\|\theta\|K)$	No
SI (Synaptic Intelligence)	$O(\|\theta\|)$	No	Scarso
Replay	GEM	$O(K \cdot B)$	Sì (sul buffer)
iCaRL	$O(K \cdot B)$	No	Buono
DER++	$O(B \cdot (d_{\text{input}} + d_{\text{logit}}))$	No	Molto buono
Architecture	Progressive Nets	$O(\|\theta\|K)$	Sì (frozen)
PackNet	$O(\|\theta\|)$ (fixed)	Sì (masked)	Sì

Certified Robustness: Randomized Smoothing

166.1 Il Problema delle Difese Euristiche

Il Adversarial Training (§54, Vol. II) è la difesa più efficace in pratica, ma fornisce solo **garanzie empiriche** — un attaccante con conoscenza del modello potrebbe trovare perturbazioni che lo ingannano. La **certified robustness** fornisce garanzie matematiche: un classificatore è (r, δ)-certifiably robust se per ogni input $\mathbf{x}$ e ogni perturbazione $\|\boldsymbol{\delta}\| \leq r$, la predizione rimane la stessa con probabilità $\geq 1 - \delta$.

166.2 Randomized Smoothing

Randomized Smoothing (Cohen et al., 2019) costruisce un **smooth classifier** g da qualsiasi base classifier f aggiungendo rumore gaussiano all'input:

$$g(\mathbf{x}) = \arg\max_{c} \mathbb{P}[f(\mathbf{x} + \boldsymbol{\varepsilon}) = c], \quad \boldsymbol{\varepsilon} \sim \mathcal{N}(\mathbf{0}, \sigma^2 \mathbf{I})$$

Teorema (Cohen et al., 2019) — Certified Radius

Sia $g(\mathbf{x}) = c_A$ con probabilità $p_A = \mathbb{P}[f(\mathbf{x}+\boldsymbol{\varepsilon}) = c_A] > 1/2$. Allora g è robusta attorno a $\mathbf{x}$ per perturbazioni ℓ_2 di raggio:

$$R = \frac{\sigma}{2}\left[\Phi^{-1}(p_A) - \Phi^{-1}(p_B)\right]$$

dove $p_B = 1 - p_A$ è la probabilità della seconda classe più probabile e Φ^{-1} è la funzione quantile della gaussiana standard.

Derivazione sketch. Il smooth classifier è robusto attorno a $\mathbf{x}$ se $g(\mathbf{x} + \boldsymbol{\delta}) = c_A$ per ogni $\|\boldsymbol{\delta}\|_2 \leq R$. Questo equivale a: $\mathbb{P}[f(\mathbf{x} + \boldsymbol{\delta} + \boldsymbol{\varepsilon}) = c_A] > 1/2$ per ogni $\boldsymbol{\delta}$ nel raggio. Usando il Neyman-Pearson lemma: la regione più difficile da difendere è quando $\boldsymbol{\delta}$ è nella direzione ottimale per il tester. La gaussiana sposta la densità di $\Phi^{-1}(p_A) - \Phi^{-1}(p_B)$ deviazioni standard prima che la probabilità di c_B superi quella di c_A.

Come stimare p_A in pratica

Per stimare p_A con garanzie statistiche, si usa un test di Clopper-Pearson (binomiale): si campionano N esempi rumorosi $\{f(\mathbf{x}+\boldsymbol{\varepsilon}_i)\}_{i=1}^{N}$, si contano i voti per c_A, e si usa il lower bound

di confidenza $1-\alpha$ come stima conservativa di p_A:

$$\hat{p}_A^- = \text{Binom-LB}(k, N, \alpha) = \text{Beta}^{-1}(\alpha; k, N-k+1)$$

Con $N = 100000$ campioni e $\alpha = 0.001$, la stima è accurata a $\pm 0.5\%$. Il certified radius usa $\hat{p}_A^-$ invece di p_A — fornisce una garanzia rigida con confidenza $1-\alpha$.

166.3 Lipschitz Networks: Robustezza Deterministica

Invece di smoothing stocastico, si costruisce direttamente una rete con costante di Lipschitz controllata. Una rete f è L-Lipschitz se $\|f(\mathbf{x}) - f(\mathbf{x}')\|_2 \leq L\|\mathbf{x} - \mathbf{x}'\|_2$ per ogni $\mathbf{x}, \mathbf{x}'$.

Per un layer lineare $f(\mathbf{x}) = \mathbf{W}\mathbf{x}$: $\|f(\mathbf{x}) - f(\mathbf{x}')\|_2 \leq \|\mathbf{W}\|_2 \|\mathbf{x} - \mathbf{x}'\|_2$ — la costante di Lipschitz è la norma spettrale $\sigma_1(\mathbf{W})$. Per una rete profonda, la costante si moltiplica lungo i layer: $L_f \leq \prod_\ell \sigma_1(\mathbf{W}_\ell)$.

SpectralNorm (§93, Vol. II) normalizza ogni layer per $\sigma_1(\mathbf{W}_\ell)$, garantendo $L_f \leq 1$ per qualsiasi profondità. Il costo: le reti 1-Lipschitz hanno capacità ridotta — c'è un tradeoff fondamentale tra robustezza L_2 e accuratezza standard (Certifiably Robust Trade-Off).

Attacchi Avversariali: Teoria e Attacchi Ottimali

167.1 PGD: Projected Gradient Descent Attack

PGD (Madry et al., 2018) è l'attacco più forte nella famiglia dei gradient-based attacks. Data una perturbazione ℓ_∞ di raggio ε, PGD trova la perturbazione che massimizza la loss:

$$\boldsymbol{\delta}^{t+1} = \Pi_{\|\boldsymbol{\delta}\|_\infty \leq \varepsilon}\big(\boldsymbol{\delta}^t + \alpha \,\text{sign}(\nabla_{\boldsymbol{\delta}} \mathcal{L}(f(\mathbf{x} + \boldsymbol{\delta}^t), y))\big)$$

dove Π è la proiezione sul ℓ_∞-ball. L'aggiornamento usa il segno del gradiente (FGSM passo) invece del gradiente grezzo — più stabile per perturbazioni ℓ_∞. Con k step e step size $\alpha = \varepsilon/k$: inizia da un punto casuale $\boldsymbol{\delta}^0 \sim \mathcal{U}[-\varepsilon, \varepsilon]^d$ per evitare vanishing gradients.

APGD: Adaptive Restart

AutoPGD (Croce e Hein, 2020) adatta automaticamente lo step size e fa restart quando la loss non migliora, senza iperparametri manuals. È la componente principale di AutoAttack

— il benchmark standard per la valutazione delle difese avversariali. Una difesa si considera "certified" solo se regge ad AutoAttack.

167.2 Adversarial Training come Minmax Optimization

L'adversarial training di Madry et al. risolve:

$$\min_{\theta} \mathbb{E}_{(\mathbf{x},y)\sim\mathcal{D}}\left[\max_{\|\boldsymbol{\delta}\|_p\leq\varepsilon} \mathcal{L}(f_\theta(\mathbf{x}+\boldsymbol{\delta}),y)\right]$$

Il problema interno (max) è risolto approssimativamente con PGD a ogni step. Il problema esterno (min) è ottimizzato con SGD usando i gradienti attraverso la perturbazione ottimale (usando il Danskin's theorem: $\nabla_\theta \max_\delta \mathcal{L}(f_\theta(\mathbf{x}+\boldsymbol{\delta}),y) = \nabla_\theta \mathcal{L}(f_\theta(\mathbf{x}+\boldsymbol{\delta}^*),y)$ dove $\boldsymbol{\delta}^*$ è la perturbazione ottimale).

Il costo computazionale: $k\times$ più costoso del training standard (k step di PGD per ogni batch). Per ridurre il costo, TRADES (Zhang et al., 2019) usa una formulazione alternativa:

$$\min_{\theta} \mathbb{E}\left[\mathcal{L}_{\text{CE}}(f_\theta(\mathbf{x}),y) + \beta \max_{\|\boldsymbol{\delta}\|\leq\varepsilon} D_{\text{KL}}(f_\theta(\mathbf{x})\|f_\theta(\mathbf{x}+\boldsymbol{\delta}))\right]$$

Il termine D_{KL} misura la stabilità del classificatore attorno a $\mathbf{x}$ — si massimizza la KL divergence tra predizione su $\mathbf{x}$ e su $\mathbf{x}+\boldsymbol{\delta}$. TRADES raggiunge il miglior tradeoff clean accuracy/robust accuracy su CIFAR-10.

Flamingo, BLIP-2 e LLaVA: Architetture Vision-Language

168.1 Flamingo: In-Context Learning per Vision-Language

Flamingo (Alayrac et al., 2022) introduce la prima architettura capace di few-shot VQA tramite in-context examples — si passano coppie (immagine, testo) come contesto nel prompt. I componenti chiave:

Perceiver Resampler: un modulo che comprime le feature visive di lunghezza variabile (da ViT) in un numero fisso di token visivi ($K = 64$). Usa cross-attention tra K query apprese e le feature visive:

$$\mathbf{X}_{\text{vis}} = \text{CrossAttn}(\underbrace{\mathbf{Q}_{\text{learned}}}_{K \times d}, \underbrace{\mathbf{F}_{\text{vision}}}_{N \times d}) \in \mathbb{R}^{K \times d}$$

Gated Cross-Attention layers (GXATTN): inseriti nei layer del LLM frozen, permettono al testo di attendere ai token visivi del Perceiver. Il gating inizializzato a zero garantisce che al training step 0, i layer visivi non interferiscano con il LLM pre-addestrato:

$$\mathbf{h}' = \mathbf{h} + \tanh(\alpha) \cdot \text{GXATTN}(\mathbf{h}, \mathbf{X}_{\text{vis}}), \quad \alpha = 0 \text{ inizialmente}$$

Solo il Perceiver Resampler e i GXATTN layers vengono addestrati — il LLM rimane frozen. Questo permette di preservare le capacità linguistiche del LLM mentre si aggiunge comprensione visiva.

168.2 BLIP-2: Querying Transformer (Q-Former)

BLIP-2 (Li et al., 2023) usa un Q-Former leggero come "ponte" tra un image encoder frozen e un LLM frozen. Il Q-Former ha un insieme di $N = 32$ query learnable che estraggono informazione visiva rilevante per il testo:

$$\mathbf{Q}_{\text{out}} = \text{QFormer}(\mathbf{Q}_{\text{learned}}, \mathbf{F}_{\text{img}}, \mathbf{T}_{\text{input}})$$

Il Q-Former combina auto-attention (tra le query stesse), cross-attention sull'immagine, e cross-attention sul testo — impara a estrarre le feature visive più rilevanti per il testo contestuale. L'output $\mathbf{Q}_{\text{out}}$ è mappato al dimensione del LLM e preposto al testo come "visual tokens".

Training in due fasi: (1) allineamento vision-language con image captioning, image-text matching, e image-grounded text generation; (2) visual-to-language generative learning agganciando il Q-Former a un LLM.

168.3 LLaVA: Large Language and Vision Assistant

LLaVA (Liu et al., 2023) usa un approccio ancora più semplice di BLIP-2: un linear projection layer che mappa le feature di CLIP al dimension del LLM. Nessun modulo speciale, solo:

$$\mathbf{V}_{\text{tokens}} = \mathbf{W} \cdot \text{CLIP-ViT}(\mathbf{I}) \in \mathbb{R}^{N_{\text{patches}} \times d_{\text{LLM}}}$$

I token visivi sono concatenati al testo e processati dall'LLM (Vicuna/LLaMA). Nonostante la semplicità, LLaVA instruction-tuned raggiunge performance competitive su benchmark VQA — la semplicità favorisce il transfer.

LLaVA-1.5 sostituisce il linear projection con un MLP a 2 layer e usa un encoder visivo più potente (CLIP ViT-L a risoluzione 336×336). Con instruction tuning su dataset misti di alta qualità, raggiunge SOTA su 11 benchmark con $< 1\%$ dei parametri di GPT-4V.

Confronto delle architetture multimodali

Modello	Vision Encoder	Bridge	LLM	Tra
Flamingo (80B)	NFNet (frozen)	Perceiver + GXATTN	Chinchilla (frozen)	Alt
BLIP-2	EVA-CLIP (frozen)	Q-Former	OPT/FlanT5 (frozen)	Me
LLaVA-1.5	CLIP ViT-L (frozen)	MLP 2-layer	Vicuna-13B (tunable)	Bas
InstructBLIP	EVA-CLIP (frozen)	Q-Former instruction-aware	Vicuna (frozen)	Me
PaLM-E (562B)	ViT-22B (jointly trained)	Linear	PaLM (jointly)	Mo

Generazione Multimodale: AudioLM, DALL-E e Modelli Unici

169.1 AudioLM: Generazione Audio come Language Modeling

AudioLM (Borsos et al., 2023) genera audio di alta qualità (parlato, musica) modellando una gerarchia di token audio. Il framework usa tre livelli di tokenizzazione:

- **Semantic tokens** (da HuBERT): catturano la struttura linguistica e il contenuto semantico — bassi bitrate, alta semantica
- **Coarse acoustic tokens** (da EnCodec): catturano l'identità vocale e le caratteristiche acustiche grossolane
- **Fine acoustic tokens** (da EnCodec): dettagli fini del waveform — alta fedeltà

La generazione è condizionata gerarchicamente: prima si generano i token semantici autoregressivamente, poi i coarse tokens condizionati sui semantici, infine i fine tokens. Questo permette di generare audio continuo coerente per molti secondi — una sfida per i modelli audio non-

gerarchici.

169.2 DALL-E 3 e la Consistenza Immagine-Testo

DALL-E 3 (Betker et al., 2023) identifica che il problema principale dei modelli text-to-image precedenti è la qualità dei caption nel dataset di training — la maggior parte dei dati web ha caption brevi, generiche e spesso non corrispondenti all'immagine.

La soluzione: si usa un modello di captioning (GPT-4V) per generare caption dettagliate sinteticamente per ogni immagine nel dataset di training — "synthetic captions" che descrivono accuratamente ogni dettaglio dell'immagine. Il modello addestrato su queste caption imita il prompt dell'utente con alta fedeltà — la consistenza testo-immagine migliora drammaticamente.

L'ablation principale: addestrare su caption umane corte vs synthetic caption lunghe — l'errore di composizionalità (es. "un cane rosso vicino a un gatto blu" predice un cane vicino a un gatto senza il coloring corretto) diminuisce di 50% con synthetic captions.

Retrieval-Augmented Generation: Teoria e Implementazione

170.1 RAG: Architettura e Componenti

RAG (Lewis et al., 2020) aumenta la generazione del LLM con informazione recuperata da un corpus esterno. L'architettura:

$$p_{\text{RAG}}(y|\mathbf{x}) = \sum_{d \in \text{TopK}(q(\mathbf{x}), p_\eta)} p_\eta(d|\mathbf{x}) \cdot p_\theta(y|\mathbf{x}, d)$$

dove $p_\eta(d|\mathbf{x}) \propto \exp(q(\mathbf{x})^T d(\mathbf{d}))$ è il retrieval score tramite dense passage retrieval (DPR), e $p_\theta(y|\mathbf{x}, d)$ è la generation likelihood condizionata sul documento d. I parametri η del retriever e θ del generator possono essere fine-tuned end-to-end.

Ragionamento sulle varianti RAG

RAG-Token vs **RAG-Sequence**: in RAG-Sequence, un unico documento viene usato per tutta la generazione della risposta. In RAG-Token, ad ogni passo di decodifica si può usare un documento diverso — più flessibile ma computazionalmente costoso. RAG-Token è superiore per compiti che richiedono combinare informazioni da fonti diverse.

170.2 Dense Passage Retrieval (DPR)

DPR (Karpukhin et al., 2020) usa due encoder BERT distinti — uno per le query $q(\mathbf{x})$ e uno per i passaggi $d(\mathbf{d})$ — addestrati con contrastive learning:

$$\mathcal{L}_{\text{DPR}} = -\log \frac{e^{q(\mathbf{x})^T d(\mathbf{d}^+)}}{e^{q(\mathbf{x})^T d(\mathbf{d}^+)} + \sum_j e^{q(\mathbf{x})^T d(\mathbf{d}_j^-)}}$$

I negativi $\mathbf{d}_j^-$ includono: (1) random negatives (passaggi casuali); (2) BM25 negatives (passaggi recuperati da TF-IDF ma non rilevanti); (3) in-batch negatives (i passaggi positivi degli altri esempi nel batch). I BM25 negatives sono "hard negatives" — particolarmente informativi perché sono lessicalmente simili ma semanticamente diversi.

170.3 ColBERT: Late Interaction per Ranking Efficiente

ColBERT (Khattab e Zaharia, 2020) usa una strategia di interazione più fine tra query e documenti. Invece di un unico embedding per documento (come DPR), ColBERT mantiene un embedding per token. Il punteggio di rilevanza:

$$S(q, d) = \sum_{i \in |\mathbf{q}|} \max_{j \in |\mathbf{d}|} \mathbf{E}_{q,i}^T \mathbf{E}_{d,j}$$

La somma sul max (MaxSim) permette ogni token della query di trovare il suo miglior match nel documento — cattura interazioni fine-grained. Le rappresentazioni dei documenti $\mathbf{E}_{d,j}$ vengono pre-calcolate e indicizzate offline; a query time, si calcola solo MaxSim con i documenti recuperati. Questo "late interaction" è 100× più efficiente del cross-encoder ma 10× più accurato di DPR.

Calibrazione dei Classificatori: ECE, Temperature Scaling e Platt Scaling

171.1 Calibrazione: Definizione Formale

Un classificatore è **perfettamente calibrato** se la probabilità predetta corrisponde alla frequenza reale di correttezza: $\mathbb{P}[\hat{y} = y | \hat{p} = p] = p$ per ogni $p \in [0, 1]$. Le reti neurali moderne sono tipicamente **overconfident** — le probabilità softmax sono maggiori della frequenza empirica di correttezza, specialmente dopo molte epoche di training.

Expected Calibration Error (ECE): si discretizzano le predizioni in M bin $\{B_m\}$ di ampiezza $1/M$ e si misura lo scarto tra confidenza media e accuracy media per ogni bin:

$$\text{ECE} = \sum_{m=1}^{M} \frac{|B_m|}{n} \left|\text{acc}(B_m) - \text{conf}(B_m)\right|$$

$$\text{acc}(B_m) = \frac{1}{|B_m|} \sum_{i \in B_m} \mathbf{1}[\hat{y}_i = y_i], \quad \text{conf}(B_m) = \frac{1}{|B_m|} \sum_{i \in B_m} \hat{p}_i$$

171.2 Temperature Scaling

Temperature Scaling (Guo et al., 2017) è il metodo di calibrazione più semplice ed efficace post-hoc. Si impara un unico parametro scalare $T > 0$ (temperatura) che scala i logit prima del softmax:

$$\hat{p}_i = \text{softmax}(\mathbf{z}_i / T)$$

$T > 1$ ammorbidisce le probabilità (riduce la confidenza — utile se overconfident); $T < 1$ le acuisce. Si ottimizza T minimizzando la Negative Log-Likelihood (NLL) su un validation set. Il parametro T non cambia l'accuratezza — solo la calibrazione.

Perché Temperature Scaling funziona (Guo et al., 2017)

Il NLL è una funzione strettamente convessa di T per qualsiasi $\mathbf{z}_i$ non costante — esiste un unico T^* ottimale. Questo T^* è il "fixing factor" che riduce l'ECE del modello. In pratica, Temperature Scaling riduce l'ECE di 5× rispetto al modello non calibrato su CIFAR-100.

171.3 Platt Scaling e Isotonic Regression

Platt Scaling (Platt, 1999) calibra un classificatore binario con una sigmoide post-hoc con due parametri (a, b) appresi su validation set:

$$\hat{p} = \sigma(a f(\mathbf{x}) + b), \quad \min_{a,b} \mathcal{L}_{\text{NLL}}$$

Isotonic Regression usa una funzione monotona non-parametrica — più flessibile di Platt ma richiede più dati. Per il caso multi-classe, si usa la **matrix scaling** (generalizzazione di temperature scaling con matrice $\mathbf{W}$ invece di scalare T): $\text{softmax}(\mathbf{W}\mathbf{z} + \mathbf{b})$.

Out-of-Distribution Detection

172.1 Il Problema OOD: Apertura a Mondo Aperto

Immaginate un sistema di diagnosi medica addestrato su immagini radiologiche di pazienti adulti, che riceve improvvisamente un'immagine pediatrica per la quale non è mai stato validato. Un classificatore tradizionale assegnerà comunque una diagnosi con alta confidenza, anche su input radicalmente diversi dalla propria distribuzione di training — comportamento noto come "overconfident predictions on OOD data" e documentato sistematicamente da Hendrycks e Gimpel (2017). Questo è il problema out-of-distribution (OOD): un modello addestrato su una distribuzione $\mathcal{D}_{\text{in}}$ deve essere capace di riconoscere che un input $\mathbf{x}$ proviene da $\mathcal{D}_{\text{out}} \neq \mathcal{D}_{\text{in}}$ — e astenersi o segnalare incertezza invece di dare una predizione sicura su una classe sbagliata. Il criterio ideale:

$$S(\mathbf{x}) \begin{cases} \text{alta} & \mathbf{x} \sim \mathcal{D}_{\text{in}} \\ \text{bassa} & \mathbf{x} \sim \mathcal{D}_{\text{out}} \end{cases}$$

La metrica standard è l'AUROC tra $S(\mathbf{x})$ sugli in-distribution e out-of-distribution samples — non dipende dalla scelta della soglia.

Maximum Softmax Probability (MSP)

La baseline di Hendrycks e Gimpel (2017): usare $S(\mathbf{x}) = \max_c \hat{p}_c(\mathbf{x})$ come score. La confidenza softmax è alta per in-distribution, bassa per OOD. Problema: le reti sono overconfident — danno spesso alta probabilità anche su input OOD.

Energy Score

Liu et al. (2020) propone lo energy score — meno sensibile alla calibrazione del softmax:

$$E(\mathbf{x}; \mathbf{f}) = -T \log \sum_c e^{f_c(\mathbf{x})/T}$$

I dati in-distribution hanno energia più bassa (maggiore densità nel feature space). Lo score OOD: $S(\mathbf{x}) = -E(\mathbf{x})$ (energia negativa = alta score per in-distribution). L'energia è proporzionale alla log-partition function della distribuzione softmax — misura la "compatibilità" dell'input con la distribuzione appresa.

Mahalanobis Distance Detector

Lee et al. (2018) usa la distanza di Mahalanobis dalle distribuzioni gaussiane fittate per classe sui feature intermedi:

$$M(\mathbf{x}) = \max_c (\mathbf{f}(\mathbf{x}) - \boldsymbol{\mu}_c)^T \boldsymbol{\Sigma}^{-1} (\mathbf{f}(\mathbf{x}) - \boldsymbol{\mu}_c)$$

dove $\boldsymbol{\mu}_c$ e $\boldsymbol{\Sigma}$ sono media di classe e covarianza pooled stimate sugli in-distribution training data. Il metodo usa feature da layer multipli e combina i punteggi di Mahalanobis tramite regressione logistica.

Bayesian Deep Learning e Uncertainty Estimation

173.1 Monte Carlo Dropout — Incertezza senza Costo Aggiuntivo

Una delle critiche più frequenti alle reti neurali in applicazioni critiche — medicina, guida autonoma, finanza — è che producono previsioni puntali senza quantificare la propria incertezza: una rete può assegnare alta confidenza a previsioni sbagliate, specialmente su dati out-of-distribution. L'inferenza bayesiana offre la soluzione teoricamente corretta — mantenere una distribuzione sui pesi $p(\mathbf{w}|\mathcal{D})$ invece di un singolo punto — ma calcolare questa distribuzione per reti profonde è computazionalmente intrattabile. MC Dropout (Gal e Ghahramani, 2016) offre una soluzione straordinariamente pratica: dimostra che il training con dropout è matematicamente equivalente all'ottimizzazione di un modello gaussiano profondo, e che mantenere il dropout attivo all'inference time produce un campionamento approssimato dalla distribuzione posterior sui pesi. Ciò significa che l'incertezza bayesiana è disponibile gratuitamente in qualsiasi rete già addestrata con dropout — senza modifiche al training. MC Dropout interpreta il dropout come approssimazione dell'inferenza bayesiana Alla test time, si mantiene il dropout attivo e si campionano T forward pass:

$$\hat{y}^{(t)} = f_\theta^{(t)}(\mathbf{x}), \quad t = 1, \ldots, T \quad \text{(dropout diverso ogni volta)}$$

$$\bar{y} = \frac{1}{T} \sum_t \hat{y}^{(t)}, \quad \text{Var}(\hat{y}) = \frac{1}{T} \sum_t (\hat{y}^{(t)} - \bar{y})^2$$

L'incertezza ha due componenti: **epistemic uncertainty** (sull'incertezza del modello — ridotta con più dati) catturata dalla varianza dei forward pass; **aleatoric uncertainty** (rumore intrinseco nei dati — irriducibile) catturata dalla varianza media delle singole predizioni.

173.2 Deep Ensembles

Deep Ensembles (Lakshminarayanan et al., 2017) addestra M reti con inizializzazioni diverse e combina le predizioni:

$$p_{\text{ens}}(y|\mathbf{x}) = \frac{1}{M}\sum_{m=1}^{M} p_{\theta_m}(y|\mathbf{x})$$

Nonostante la semplicità, Deep Ensembles produce le incertezze più calibrate e rileva meglio gli OOD rispetto a metodi bayesiani più sofisticati. Il motivo: ogni rete esplora un bacino di attrazione diverso nello spazio dei pesi — la diversità delle soluzioni è più informativa dell'incertezza stimata da un singolo modello perturbato.

Il costo computazionale è $M\times$ quello di una singola rete — il limite principale. **Batch Ensembles** (Wen et al., 2020) riduce il costo memorizzando solo un rank-1 perturbation per membro: $\mathbf{W}_m = \mathbf{W} \odot (\mathbf{r}_m \mathbf{s}_m^T)$ — costo di memoria aggiuntivo $O(Md)$ invece di $O(M|\theta|)$.

AdaFactor, Lion e Muon: Ottimizzatori Moderni

174.1 AdaFactor: Memoria O($\sqrt{n}$) per Matrici Grandi

Adam richiede memorizzare momenti di primo e secondo ordine per ogni parametro — $2|\theta|$ storage aggiuntivo. Per LLM con miliardi di parametri, questo è proibitivo. AdaFactor (Shazeer e Stern, 2018) fattorizza il secondo momento per matrici di pesi $\mathbf{W} \in \mathbb{R}^{m\times n}$:

$$\hat{V}_{t,ij} \approx \frac{R_{t,i} C_{t,j}}{S_t}, \quad R_{t,i} = \sum_j V_{t,ij}, \quad C_{t,j} = \sum_i V_{t,ij}, \quad S_t = \sum_{ij} V_{t,ij}$$

Invece di memorizzare $V \in \mathbb{R}^{m\times n}$, si memorizzano $R \in \mathbb{R}^m$ e $C \in \mathbb{R}^n$ — riduzione da $O(mn)$ a $O(m+n)$. AdaFactor usa questo secondo momento fattorizzato per normalizzare i gradienti, con moment di primo ordine disabilitato per default (solo quando $d < 10^4$).

174.2 Lion: Evolved Optimizer

Lion (Chen et al., 2023) è un ottimizzatore scoperto tramite program search (evoluzione di espressioni simboliche). L'aggiornamento sorprendentemente semplice:

$$\mathbf{c}_t = \beta_1 \mathbf{m}_{t-1} + (1-\beta_1)\mathbf{g}_t \qquad \text{(interpolazione per update direction)}$$

$$\theta_t = \theta_{t-1} - \eta \cdot \text{sign}(\mathbf{c}_t) \qquad \text{(sign update — trust region implicita)}$$

$$\mathbf{m}_t = \beta_2 \mathbf{m}_{t-1} + (1-\beta_2)\mathbf{g}_t \qquad \text{(aggiorna momentum per il prossimo step)}$$

L'update è sempre $\pm\eta$ — la norma dell'aggiornamento è costante, indipendente dalla magnitudine del gradiente. Questo è equivalente a un trust-region implicito con raggio η nella norma ℓ_∞. Lion usa meno memoria di Adam (un solo momento invece di due) ed è 1.5–2.5× più efficiente in compute per step comparabile.

174.3 Muon: Ottimizzazione in Spazio di Nesterov

Muon (Kosson et al., 2024) applica l'aggiornamento di Nesterov nei *layer weights* proiettando il gradiente ortogonalizzato tramite Newton-Schulz:

$$\tilde{G} = \text{Orthogonalize}(G) \quad \text{s.t.} \quad \tilde{G}^T\tilde{G} = \mathbf{I}$$

L'ortogonalizzazione tramite Newton-Schulz (5 iterazioni convergenti): $X_{k+1} = X_k(3\mathbf{I} - X_k^T X_k)/2$ partendo da $X_0 = G/\|G\|_F$. Il gradiente ortogonalizzato garantisce che ogni direzione di peso riceva un aggiornamento di pari magnitudine — risolve il problema del "gradient imbalance" dove alcuni pesi ricevono aggiornamenti molto più grandi di altri.

174.4 Learning Rate Schedules: Analisi Comparativa

Schedule	Formula η_t	Proprietà
Cosine Annealing	$\eta_{\min} + \frac{1}{2}(\eta_{\max} - \eta_{\min})(1 + \cos(\pi t/T))$	Smooth, molto usato per LLM
Warmup + Decay	$\eta_{\max} \cdot \min(t/T_w, (T-t)/(T-T_w))$	Warmup evita instabilità inizial
Linear Decay	$\eta_0(1 - t/T)$	Semplice, usato in LLM pre-tra
1-cycle Policy	Warmup fino a $\eta_{\max}$, poi cosine decay	Super-convergence in poche epo
Constant	η_0	Semplice ma spesso subottimale
Inverse Square Root	$\eta_0/\sqrt{t}$	Originale Transformer — teoria
WSD (Warmup-Stable-Decay)	3 fasi separate	Ottimale per pre-training conti

Il **Warmup-Stable-Decay (WSD)** schedule (Hu et al., 2024) è particolarmente interessante per il continual pre-training: la fase "stable" permette di aggiungere dati liberamente, e la fase "decay" converte le ultime iterazioni in un fine-tuning implicito. Si può anche fare "branch" dallo stesso stable checkpoint a diversi decay paths per creare specializzazioni del modello.

Gradient Checkpointing, Mixed Precision e Ottimizzazioni di Sistema

175.1 Gradient Checkpointing: Tradeoff Memoria-Compute

Il training standard memorizza tutte le attivazioni intermedie per il backward pass — costo memoria $O(L)$ per layer L profondi. Il **gradient checkpointing** (Chen et al., 2016) ri-calcola le attivazioni durante il backward invece di memorizzarle:

$$\text{Memoria standard: } O(L \cdot B \cdot d) \quad \rightarrow \quad \text{Con checkpointing: } O(\sqrt{L} \cdot B \cdot d)$$

La strategia ottimale: si mantengono $\sqrt{L}$ checkpoint (attivazioni salvate) equidistribuiti; tra checkpoint adiacenti si ri-calcolano al volo durante il backward. Il costo extra di compute è $O(\sqrt{L})$ forward pass aggiuntivi — tradeoff accettabile.

175.2 Mixed Precision Training: BF16 vs FP16

Il training in mezza precisione usa FP16 o BF16 per forward/backward e FP32 per l'aggiornamento dei pesi (master weights). La differenza chiave tra i due formati:

Formato	Bit (exp/mantissa)	Range	Precisione	Preferito per
FP32	8/23	$\pm 3.4 \times 10^{38}$	~ 7 cifre	Master weights, ottimizzatore
FP16	5/10	± 65504	~ 3.3 cifre	GPU Nvidia Volta/Turing
BF16	8/7	$\pm 3.4 \times 10^{38}$	~ 2.3 cifre	TPU, GPU Ampere+ (preferito)
FP8 (E4M3)	4/3	± 448	~ 1.2 cifre	Forward pass H100

BF16 è preferito a FP16 per i LLM perché ha lo stesso range di FP32 — non richiede loss scaling (tecnica che scala il loss prima del backward per evitare underflow in FP16). L'overflow di FP16 (gradiente > 65504) è un problema comune nella pratica che non esiste con BF16.

175.3 Activation Offloading e Recomputation in Pipeline

In pipeline parallelism, ogni stage deve memorizzare le attivazioni dei micro-batch che sono "in volo" nella pipeline. Con K stage e M micro-batch, ogni stage deve memorizzare fino a $K-1$ set di attivazioni simultaneamente — il bottleneck di memoria della pipeline.

La strategia **activation offloading**: le attivazioni vengono spostate da GPU a CPU RAM non appena vengono calcolate, e riportate sulla GPU solo quando servono per il backward. Con NVLink o PCIe-4 (100 GB/s), il trasferimento per un LLM standard è nell'ordine dei millisecondi — accettabile se i forward pass successivi "coprono" il trasferimento (compute-communication overlap).

Glossario Completo dei Simboli del Volume III

A.1 Spazi, Distribuzioni e Probabilità

Simbolo	Significato
$\mathcal{X}, \mathcal{Y}, \mathcal{Z}$	Spazio di input, output, latente
$p_{\text{data}}(\mathbf{x}), p_G(\mathbf{x})$	Distribuzione reale e distribuzione generata (GAN)
$p(\mathbf{z}), q_\phi(\mathbf{z}\|\mathbf{x})$	Prior e posterior variazionale (VAE)
$p_\theta(\mathbf{x}\|\mathbf{z})$	Likelihood del decoder (VAE)
$\bar{\alpha}_t = \prod_{s=1}^{t} \alpha_s$	Cumulative noise schedule (DDPM)
$q(\mathbf{x}_t\|\mathbf{x}_0)$	Marginal forward process (DDPM) $= \mathcal{N}(\sqrt{\bar{\alpha}_t}\mathbf{x}_0, (1-\bar{\alpha}_t)\mathbf{I})$
$\tilde{\boldsymbol{\mu}}_t, \tilde{\beta}_t$	Media e varianza della reverse posterior trattabile (DDPM)
$\boldsymbol{\varepsilon}_\theta(\mathbf{x}_t, t)$	Noise-predicting network (DDPM/DDIM)
τ (temperature contrastive)	Temperatura nella NT-Xent / InfoNCE loss
$\text{sim}(\mathbf{u}, \mathbf{v})$	Cosine similarity: $\mathbf{u}^T\mathbf{v}/(\|\mathbf{u}\|\|\mathbf{v}\|)$
$\Pi(P, Q)$	Insieme dei joint distributions con marginali P, Q (OT)
$W_1(P, Q)$	Wasserstein-1 distance tra P e Q
$\text{JSD}(P\|Q)$	Jensen-Shannon Divergence
$p_A = \mathbb{P}[f(\mathbf{x}+\boldsymbol{\varepsilon}) = c_A]$	Probabilità della classe A sotto smoothing (Randomized Smoothing)
$R = \frac{\sigma}{2}[\Phi^{-1}(p_A) - \Phi^{-1}(p_B)]$	Certified radius dello smooth classifier

A.2 Reinforcement Learning

Simbolo	Significato
$(\mathcal{S}, \mathcal{A}, P, R, \gamma)$	Tupla MDP: stati, azioni, transizione, reward, discou
$\pi : \mathcal{S} \to \Delta(\mathcal{A})$	Policy stocastica (distribuzione sulle azioni)
$V^\pi(s), Q^\pi(s,a), A^\pi(s,a)$	State-value, action-value, advantage function
$G_t = \sum_{k \geq 0} \gamma^k R_{t+k+1}$	Discounted return dal passo t
$\delta_t = R_{t+1} + \gamma V(S_{t+1}) - V(S_t)$	TD error (Temporal Difference)
$d^\pi(s)$	Distribuzione stazionaria degli stati sotto π
$\hat{A}_t^{\text{GAE}(\lambda)} = \sum_{l \geq 0} (\gamma\lambda)^l \delta_{t+l}$	GAE advantage con parametro λ
$r_t(\theta) = \pi_\theta(a_t \mid s_t) / \pi_{\theta_{\text{old}}}(a_t \mid s_t)$	Importance sampling ratio (PPO)
$\varepsilon_{\text{clip}}$	Clipping range in PPO (tipicamente 0.1–0.2)
$H(\pi(\cdot \mid s)) = -\mathbb{E}_{a \sim \pi}[\log \pi(a \mid s)]$	Entropia della policy (SAC)
α (SAC)	Temperatura maximum entropy RL
$V^\pi_{\text{soft}}(s) = \mathbb{E}_\pi[Q^\pi_{\text{soft}} - \alpha \log \pi]$	Soft value function (SAC)
$\mathcal{D}$ (replay buffer)	Buffer di transizioni (s, a, r, s') per experience replay
θ^-	Parametri della target network (DQN, aggiornata og
$\rho^\pi = \lim_{T \to \infty} \frac{1}{T} \mathbb{E}_\pi[\sum_{t<T} R_t]$	Average reward criterion
$\rho^* + V^*(s) = \max_a [R(s,a) + \sum_{s'} P(s' \mid s, a) V^*(s')]$	Bellman relative value equation

A.3 Meta-Learning e Self-Supervised

Simbolo	Significato
$\mathcal{T}, p(\mathcal{T})$	Task e distribuzione sui task
$\mathcal{S}, \mathcal{Q}$	Support set e query set (few-shot)
θ', α (MAML)	Parametri adattati e inner learning rate
$\mathbf{c}_n = \frac{1}{K} \sum_{\mathbf{x} \in \mathcal{S}_n} f_\theta(\mathbf{x})$	Prototipo della classe n (Prototypical Networks)
$\mathbf{z}_{\text{online}}, \mathbf{z}_{\text{target}}$	Proiezioni delle due reti in BYOL
$\mathbf{c} \leftarrow m\mathbf{c} + (1-m) g_t(\mathbf{x})$	Centering EMA del teacher (DINO)
$\xi \leftarrow \tau\xi + (1-\tau)\theta$	EMA update della target network (MoCo, BYOL)
$\mathcal{M}$ (MAE)	Insieme delle patch mascherate
$\mathcal{C}_{ij}$	Cross-correlation matrix (Barlow Twins)

A.4 Teoria Statistica

Simbolo	Significato
$d_{\mathrm{VC}}(\mathcal{H})$	VC dimension della classe $\mathcal{H}$
$\Pi_{\mathcal{H}}(m)$	Growth function: max dicotomie su m punti
$\mathfrak{R}_m(\mathcal{H}), \hat{\mathfrak{R}}_S(\mathcal{H})$	Rademacher complexity (popolazione e empirica)
$\sigma_i \sim \mathrm{Uniform}(\{-1,+1\})$	Variabile di Rademacher
β (stabilità uniforme)	$\sup_z \lvert\ell(A(S),z) - \ell(A(S^i),z)\rvert \leq \beta$
$D_{\mathrm{KL}}(Q\Vert P)$	KL divergence nella PAC-Bayes bound
$\lambda_1 \geq \lambda_2 \geq \cdots$	Autovalori della covarianza (double descent)
$k_n = \sum_{k>n} \lambda_k / \lambda_n$	Dimensione effettiva della coda (benign overfitting)
$R_T = T\mu^* - \sum_t \mathbb{E}[r_t]$	Pseudo-regret del bandit (MAB)
$\Delta_k = \mu^* - \mu_k$	Gap suboptimality dell'arm k

A.5 Training Distribuito e Inferenza Efficiente

Simbolo	Significato
K (worker/GPU)	Numero di device nel sistema distribuito
Ψ (parametri)	Numero totale di parametri del modello
s (scale), z_{point}	Scale factor e zero-point nella quantizzazione uniforme
b (bit-width)	Larghezza in bit per la quantizzazione ($b = 4, 8$)
$\mathbf{H} = \mathbf{X}\mathbf{X}^T$	Hessiana (Gram matrix) per GPTQ
$\mathcal{I}_k = \lvert\partial\mathcal{L}/\partial s_k\rvert \cdot \lvert s_k\rvert$	Importance score per il pruning strutturato
T_{dist} (distillation)	Temperatura della soft loss nella knowledge distillation
$\mathbf{S}$ (Flash Attention)	Running softmax statistics (max, normalization factor)
M (SRAM size)	Dimensione della cache on-chip per Flash Attention tiling
$\hat{V}_{t,ij} \approx R_{t,i} C_{t,j} / S_t$	Secondo momento fattorizzato (AdaFactor)
$\mathrm{sign}(\cdot)$	Funzione segno nell'update di Lion

A.6 Vision, Time Series e Multimodale

Simbolo	Significato
$\hat{\sigma}$ (DETR)	Assegnamento ottimale Hungarian matching
$\mathcal{C}_{\text{match}}(y_i, \hat{y}_j)$	Costo di abbinamento DETR (class + box)
N_{queries}	Numero di object queries in DETR (tipicamente 100)
L, H, C (time series)	Lookback length, horizon, canali (variabili)
P, S (PatchTST)	Patch size e stride per tokenizzazione temporale
$N_{\text{patch}} = \lfloor (L-P)/S \rfloor + 2$	Numero di patch temporali
$\hat{\boldsymbol{\theta}}_f, \hat{\boldsymbol{\theta}}_b$	Coefficienti forecast/backcast N-BEATS
$\mathbf{V}_f$ (N-BEATS)	Basis matrix (Fourier o polinomiale)
$q(\mathbf{x}), d(\mathbf{d})$	Query e document embedding (DPR)
$S(q,d) = \sum_i \max_j \mathbf{E}_{q,i}^T \mathbf{E}_{d,j}$	MaxSim ColBERT score
$\mathbf{Q}_{\text{learned}}$ (BLIP-2)	Query learnable del Q-Former ($N = 32$ query)

Tabelle di Complessità Computazionale

B.1 Modelli Generativi — Costo di Training e Sampling

Modello	Training (per step)	Sampling (per immagine)	Memoria
VAE	$O(B \cdot d_z)$ + forward decoder	$O(d_z)$ + forward	$O(\lvert\theta\rvert + Bd_z)$
GAN	$O(B)$ per G e D	$O(d_z)$ + 1 forward G	$O(\lvert\theta_G\rvert + \lvert\theta_D\rvert)$
NICE/RealNVP	$O(B \cdot d)$ — det gratuito	$O(Ld)$ (L coupling layers)	$O(\lvert\theta\rvert)$ — nessuna m
DDPM	$O(B)$ — 1 denoising step	$O(T)$ step sequenziali	$O(\lvert\theta_{\text{UNet}}\rvert)$
DDIM	Come DDPM	$O(S)$ step, $S \ll T$	Come DDPM
Consistency Model	Distillazione da solver	$O(1)$–$O(k)$ step	$O(\lvert\theta\rvert)$

B.2 Attention Variants — Costo in N (lunghezza sequenza) e d (dimensione)

Metodo	Training compute	Training memoria	Inferenza per to
Full Attention	$O(N^2 d)$	$O(N^2 + Nd)$	$O(Nd)$ (KV cache
Flash Attention	$O(N^2 d)$ — stessi FLOPs	$O(N)$ — no materialization	$O(Nd)$
Linear Attention	$O(Nd^2)$	$O(Nd + d^2)$	$O(d^2)$ — RNN-like
Longformer (window w)	$O(Nwd)$	$O(Nw)$	$O(wd)$
Swin (window M)	$O(NM^2 d/M^2) = O(Nd)$	$O(NM^2 + Nd)$	$O(Nd/N_{\text{heads}})$
Ring Attention (K device)	$O(N^2 d/K)$ per device	$O(N^2/K + Nd)$	$O(Nd)$
Mamba SSM	$O(Nd^2)$ (parallel scan)	$O(Nd)$	$O(d^2)$ — RNN

B.3 Reinforcement Learning — Complessità degli Algoritmi

Algoritmo	Aggiornamento per step	Memoria	Sample efficiency
Q-Learning tabellare	$O(1)$	$O(\|\mathcal{S}\|\|\mathcal{A}\|)$	Bassa
DQN	$O(B \cdot d_{\text{NN}})$	$O(\|\theta\| + B_{\text{replay}})$	Media
REINFORCE	$O(T \cdot d_{\text{NN}})$ per traiettoria	$O(\|\theta\| + T)$	Bassa (alta varianza)
PPO	$O(K_{\text{epoch}} \cdot B \cdot d_{\text{NN}})$	$O(\|\theta\| + B \cdot T)$	Media-alta
SAC	$O(B \cdot d_{\text{NN}})$ — 3 reti	$O(3\|\theta\| + B_{\text{replay}})$	Alta (off-policy)
MAML (inner loop)	$O(k \cdot \|\theta\|^2)$ second-order	$O(k \cdot \|\theta\|)$ per task	Alta (pochi esempi)

B.4 Training Distribuito — Riduzione di Memoria con ZeRO

Configurazione	Memoria per GPU	Comunicazione aggiuntiva
Baseline (DP standard)	16Ψ bytes	AllReduce gradient: $2(K-1)\Psi/K$
ZeRO Stage 1	$4\Psi + 12\Psi/K$ bytes	Come baseline
ZeRO Stage 2	$4\Psi + 2\Psi/K + 12\Psi/K$ bytes	ReduceScatter + AllReduce
ZeRO Stage 3	$16\Psi/K$ bytes	AllGather params: $\sim 3\Psi$
ZeRO-Infinity	≈ 0 su GPU	NVLink/PCIe CPU offload
Tensor Parallelism (K)	$\|\theta\|/K$ per layer	2 AllReduce per Transformer block

Identità Matematiche Fondamentali del Volume III

C.1 VAE, GAN e Normalizing Flows

VAE	
$\mathcal{L}_{\text{ELBO}} = \mathbb{E}_{q_\phi}[\log p_\theta(\mathbf{x}\|\mathbf{z})] - D_{\text{KL}}(q_\phi\|p)$	ELBO — lower bound su $\log p(\mathbf{x})$
$\log p(\mathbf{x}) - \mathcal{L}_{\text{ELBO}} = D_{\text{KL}}(q_\phi(\mathbf{z}\|\mathbf{x})\|p(\mathbf{z}\|\mathbf{x})) \geq 0$	Gap = KL posteriore
$D_{\text{KL}}(\mathcal{N}(\boldsymbol{\mu}, \text{diag}(\boldsymbol{\sigma}^2))\|\mathcal{N}(\mathbf{0}, \mathbf{I})) = -\frac{1}{2}\sum_j(1 + \log\sigma_j^2 - \mu_j^2 - \sigma_j^2)$	KL in forma chiusa
$\mathbf{z} = \boldsymbol{\mu}_\phi + \boldsymbol{\sigma}_\phi \odot \boldsymbol{\varepsilon}$, $\boldsymbol{\varepsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$	Reparametrizzazione
GAN	
$D_G^*(\mathbf{x}) = p_{\text{data}}(\mathbf{x})/(p_{\text{data}}(\mathbf{x}) + p_G(\mathbf{x}))$	Discriminatore ottimale
$\min_G V(G, D_G^*) = -\log 4 + 2\,\text{JSD}(p_{\text{data}}\|p_G)$	Equilibrio GAN = minimax di JSD
$W_1(P, Q) = \sup_{\|f\|_L \leq 1}[\mathbb{E}_P[f] - \mathbb{E}_Q[f]]$	Dualità Kantorovich-Rubinstein
Normalizing Flows	
$\log p(\mathbf{x}) = \log p_Z(f(\mathbf{x})) + \log\|\det \mathbf{J}_f(\mathbf{x})\|$	Change of variables
$\log\|\det \mathbf{J}\| = \sum_{i>k} s_i(x_{1:k})$ per affine coupling	RealNVP — costo $O(d)$
$\log\|\det \mathbf{J}\| = H \cdot W \sum_c \log\|s_c\|$ per ActNorm	Glow ActNorm layer
DDPM	
$\mathbf{x}_t = \sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\varepsilon}$, $\boldsymbol{\varepsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$	Forward marginal in 1 step
$\tilde{\boldsymbol{\mu}}_t = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\varepsilon}\right)$	Reverse posterior mean
$\mathcal{L}_{t-1} \propto \|\boldsymbol{\varepsilon} - \boldsymbol{\varepsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\varepsilon}, t)\|^2$	DDPM training objective

C.2 Reinforcement Learning

Equazioni fondamentali	
$V^\pi(s) = \sum_a \pi(a\|s)[R(s,a) + \gamma\sum_{s'} P(s'\|s,a)V^\pi(s')]$	Bellman expectation
$V^*(s) = \max_a[R(s,a) + \gamma\sum_{s'} P(s'\|s,a)V^*(s')]$	Bellman optimality
$Q(S_t,A_t) \leftarrow Q(S_t,A_t) + \alpha[\underbrace{R_{t+1} + \gamma\max_{a'} Q(S_{t+1},a')}_{\text{TD target}} - Q(S_t,A_t)]$	Q-Learning update
$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}[\nabla_\theta \log\pi_\theta(A\|S)\cdot Q^{\pi_\theta}(S,A)]$	Policy Gradient Theorem
$\mathcal{L}^{\text{CLIP}} = \mathbb{E}_t[\min(r_t\hat{A}_t, \text{clip}(r_t, 1-\varepsilon, 1+\varepsilon)\hat{A}_t)]$	PPO clipped objective
$\pi^*_{\text{soft}}(a\|s) \propto \exp(Q^*_{\text{soft}}(s,a)/\alpha)$	SAC optimal policy
$\mathcal{L}^{\text{RLHF}}_{\text{RM}} = -\mathbb{E}_{(x,y_1\succ y_2)}[\log\sigma(r_\phi(x,y_1) - r_\phi(x,y_2))]$	Reward model (Bradley-Terry)
$\max_\theta \mathbb{E}[r_\phi(x,y)] - \beta D_{\text{KL}}(\pi_\theta\|\pi_{\text{SFT}})$	RLHF PPO objective

C.3 Teoria Statistica

VC, Rademacher, PAC	
$\Pi_{\mathcal{H}}(m) \le \sum_{i=0}^{d} \binom{m}{i} \le (em/d)^d$	Lemma di Sauer-Shelah
$R(h) \le \hat{R}_m(h) + 2\mathfrak{R}_m(\mathcal{H}) + \sqrt{\ln(1/\delta)/2m}$	Bound Rademacher (Koltchinskii-Panchenko)
$\mathfrak{R}_m(\mathcal{H}_B) \le BR/\sqrt{m}$ per $\|\mathbf{w}\| \le B$, $\|\mathbf{x}\| \le R$	Rademacher per iperpiani
$m_{\mathcal{H}}(\varepsilon, \delta) = O\left(\frac{d \log(1/\varepsilon) + \log(1/\delta)}{\varepsilon^2}\right)$	Sample complexity PAC
$R(\hat{f}_m) \le O\left(\sqrt{\|\mathbf{y}\|^2_{K_\infty^{-1}}/m}\right)$	Generalizzazione NTK (norma RKHS)
$R(\hat{\mathbf{w}}) \approx \sigma^2 \frac{n}{d-n-1} + \|\mathbf{w}^*\|^2(1 - n/d)$	Double descent — regressione lineare
$\mathbb{E}_{h\sim Q}[R(h)] \le \hat{R}_m + \sqrt{(D_{\mathrm{KL}}(Q\|P) + \ln(1/\delta))/(2m)}$	PAC-Bayes (McAllester)
$R_T \ge \Omega(\sum_{k:\Delta_k>0} \log T/\Delta_k)$ per MAB	Lower bound regret Lai-Robbins

C.4 Calibrazione, OOD e Robustezza

Calibrazione e uncertainty	
$\mathrm{ECE} = \sum_m \frac{\|B_m\|}{n} \|\mathrm{acc}(B_m) - \mathrm{conf}(B_m)\|$	Expected Calibration Error
$\hat{p} = \mathrm{softmax}(\mathbf{z}/T^*)$, $T^* = \arg\min_T \mathrm{NLL}_{\mathrm{val}}$	Temperature Scaling
$E(\mathbf{x}) = -T \log \sum_c e^{f_c(\mathbf{x})/T}$	Energy score per OOD detection
$M(\mathbf{x}) = \max_c (\mathbf{f}(\mathbf{x}) - \boldsymbol{\mu}_c)^T \boldsymbol{\Sigma}^{-1} (\mathbf{f}(\mathbf{x}) - \boldsymbol{\mu}_c)$	Mahalanobis OOD detector
Adversarial	
$\boldsymbol{\delta}^{t+1} = \Pi_{\|\boldsymbol{\delta}\|_\infty \le \varepsilon}(\boldsymbol{\delta}^t + \alpha \mathrm{sign}(\nabla_{\boldsymbol{\delta}} \mathcal{L}))$	PGD attack step
$\mathcal{L}_{\mathrm{TRADES}} = \mathcal{L}_{\mathrm{CE}} + \beta \max_{\|\boldsymbol{\delta}\| \le \varepsilon} D_{\mathrm{KL}}(f(\mathbf{x}) \| f(\mathbf{x} + \boldsymbol{\delta}))$	TRADES adversarial training
$R = \frac{\sigma}{2}[\Phi^{-1}(p_A) - \Phi^{-1}(1 - p_A)]$	Certified radius (binary case)
$\hat{p}_A^- = \mathrm{Beta}^{-1}(\alpha; k, N - k + 1)$	Clopper-Pearson lower bound (certified)

Diagnostica Completa e Iperparametri Consigliati

D.1 Checklist di Debug — Guida Sistematica

Fase	Sintomo	Causa probabile
Generativi	Immagini VAE sfocate	KL troppo alto, decoder deb
GAN mode collapse	Discriminatore troppo forte / JSD piatta	WGAN-GP, spectral norm,
DDPM rumore residuo	T insufficiente, schedule sbagliato	Usare cosine schedule, aume
RL	Policy collassa dopo miglioramento	KL troppo grande in un upd
Q-values esplodono	Reward non scalato, gradiente unstable	Clipping reward, gradient cl
SAC non esplora	Temperatura α troppo bassa	Abilitare auto-tuning di α
RLHF reward hacking	Reward model overfit / β KL troppo basso	Aumentare β, più dati RM
SSL	Rappresentazioni costanti (collapse)	Mancano negativi / asimme
Linear probe scarsa con SSL	Augmentation troppo aggressiva	Ridurre intensità, crop più g
MAML non adatta	Inner LR troppo alto/basso	Meta-SGD per LR per-para
Distribuito	NaN durante training FP16	Overflow FP16
OOM nonostante ZeRO	Attivazioni non sharded	Aggiungere gradient checkp
Speedup deludente con DP	Comunicazione domina su compute	Aumentare batch size, usare
Calibrazione	Alta confidenza su OOD	Overconfidence post-softmax
ECE alta dopo training	Overfit al training set	Temperature scaling su val s

D.2 Iperparametri Consigliati — Tabella Estesa

Modello	Iperparametro	Valore tipico	Note
DDPM	T (timestep totali)	1000	Cosine schedule rac
$\beta_{\min}, \beta_{\max}$	0.0001, 0.02 (linear)	Cosine: $\beta_t = 1 - \cos(\pi t/T)^2$	
Architettura UNet	Base dim = 128 o 256	Attention a 16×16 e 8×8	
CFG scale w	7.5	Range utile: 3–15	
PPO	ε (clip)	0.2	Abbassare per alta
GAE λ	0.95	0.9–0.99 a seconda del rumore	
Entropy coeff	0.01	Aumentare se policy collassa presto	
Epochs per rollout	4–10	Più epoche = più compute efficiente	
SAC	Replay buffer	10^6	Aumentare per tasl
Soft update τ	0.005	5×10^{-3} stabile	
Target entropy	$-\lvert\mathcal{A}\rvert$	Per azioni continue	
SimCLR/BYOL	Temperatura τ	0.07 (SimCLR), N/A (BYOL)	Più bassa = più co
Projection dim	128–256	Più grande migliora ma costoso	
EMA τ (BYOL)	0.996–0.999	Aumentare con batch size	
MAML	Inner LR α	0.01	Grid search su 0.00
Inner steps	5	1 per FOMAML veloce	
Meta batch size	4–32 task	Bilanciare varietà e compute	
MoE	E (expert)	8–64	Switch: $E = 128+$
k (top-k)	1 (Switch), 2 (Mixtral)	$k = 1$ max efficienza	
α (aux loss)	0.01	Aumentare se load imbalanced	
ZeRO	Stage	2 o 3	Stage 3 per modelli
Offload	Disabilitato	Abilitare solo se necessario (lento)	
Calibrazione	Temperatura T	Ottimizzare su val set	Range tipico: 1–5
Bin ECE M	15	Standard in letteratura	

Bibliografia Completa del Volume III

E.1 Modelli Generativi — Riferimenti

- Kingma, D.P., Welling, M. (2014). *Auto-Encoding Variational Bayes.* ICLR 2014.
- Higgins, I., et al. (2017). *β-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework.* ICLR 2017.
- Vahdat, A., Kautz, J. (2020). *NVAE: A Deep Hierarchical Variational Autoencoder.* NeurIPS 2020.

- Goodfellow, I., et al. (2014). *Generative Adversarial Nets.* NeurIPS 2014.
- Arjovsky, M., Chintala, S., Bottou, L. (2017). *Wasserstein GAN.* ICML 2017.
- Gulrajani, I., et al. (2017). *Improved Training of Wasserstein GANs.* NeurIPS 2017.
- Karras, T., Laine, S., Aila, T. (2019). *A Style-Based Generator Architecture for GANs.* CVPR 2019.
- Dinh, L., Krueger, D., Bengio, Y. (2015). *NICE: Non-linear Independent Components Estimation.* ICLR Workshop 2015.
- Dinh, L., Sohl-Dickstein, J., Bengio, S. (2017). *Density Estimation using Real NVP.* ICLR 2017.
- Kingma, D.P., Dhariwal, P. (2018). *Glow: Generative Flow with Invertible 1×1 Convolutions.* NeurIPS 2018.
- Ho, J., Jain, A., Abbeel, P. (2020). *Denoising Diffusion Probabilistic Models.* NeurIPS 2020.
- Song, J., Meng, C., Ermon, S. (2021). *Denoising Diffusion Implicit Models.* ICLR 2021.
- Song, Y., et al. (2023). *Consistency Models.* ICML 2023.
- Rombach, R., Blattmann, A., Lorenz, D., Esser, P., Ommer, B. (2022). *High-Resolution Image Synthesis with Latent Diffusion Models.* CVPR 2022.
- Ho, J., Salimans, T. (2022). *Classifier-Free Diffusion Guidance.* NeurIPS Workshop 2021.
- Betker, J., et al. (2023). *Improving Image Generation with Better Captions.* OpenAI Technical Report.

E.2 Reinforcement Learning — Riferimenti

- Sutton, R.S., Barto, A.G. (2018). *Reinforcement Learning: An Introduction, 2nd ed.* MIT Press.
- Watkins, C.J.C.H., Dayan, P. (1992). *Q-Learning.* Machine Learning 8.
- Mnih, V., et al. (2015). *Human-Level Control through Deep Reinforcement Learning.* Nature 518.
- van Hasselt, H., Guez, A., Silver, D. (2016). *Deep Reinforcement Learning with Double*

Q-learning. AAAI 2016.

- Schaul, T., Quan, J., Antonoglou, I., Silver, D. (2016). *Prioritized Experience Replay.* ICLR 2016.
- Bellemare, M.G., Dabney, W., Munos, R. (2017). *A Distributional Perspective on Reinforcement Learning.* ICML 2017.
- Hessel, M., et al. (2018). *Rainbow: Combining Improvements in Deep Reinforcement Learning.* AAAI 2018.
- Williams, R.J. (1992). *Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning.* Machine Learning.
- Sutton, R.S., McAllester, D., Singh, S., Mansour, Y. (2000). *Policy Gradient Methods for Reinforcement Learning with Function Approximation.* NeurIPS 1999.
- Schulman, J., et al. (2016). *High-Dimensional Continuous Control Using Generalized Advantage Estimation.* ICLR 2016.
- Schulman, J., et al. (2017). *Proximal Policy Optimization Algorithms.* arXiv:1707.06347.
- Haarnoja, T., et al. (2018). *Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning.* ICML 2018.
- Kumar, A., et al. (2020). *Conservative Q-Learning for Offline Reinforcement Learning.* NeurIPS 2020.
- Chen, L., et al. (2021). *Decision Transformer: Reinforcement Learning via Sequence Modeling.* NeurIPS 2021.
- Reed, S., et al. (2022). *A Generalist Agent (Gato).* TMLR 2022.
- Christiano, P., et al. (2017). *Deep Reinforcement Learning from Human Preferences.* NeurIPS 2017.
- Ouyang, L., et al. (2022). *Training Language Models to Follow Instructions with Human Feedback.* NeurIPS 2022.

E.3 Self-Supervised, Meta-Learning e Teoria

- van den Oord, A., Li, Y., Vinyals, O. (2018). *Representation Learning with Contrastive Predictive Coding.* arXiv:1807.03748.

- Chen, T., et al. (2020). *A Simple Framework for Contrastive Learning of Visual Representations.* ICML 2020.
- He, K., et al. (2020). *Momentum Contrast for Unsupervised Visual Representation Learning.* CVPR 2020.
- Grill, J.B., et al. (2020). *Bootstrap Your Own Latent: A New Approach to Self-Supervised Learning.* NeurIPS 2020.
- He, K., et al. (2022). *Masked Autoencoders Are Scalable Vision Learners.* CVPR 2022.
- Caron, M., et al. (2021). *Emerging Properties in Self-Supervised Vision Transformers (DINO).* ICCV 2021.
- Zbontar, J., et al. (2021). *Barlow Twins: Self-Supervised Learning via Redundancy Reduction.* ICML 2021.
- Gao, T., Yao, X., Chen, D. (2021). *SimCSE: Simple Contrastive Learning of Sentence Embeddings.* EMNLP 2021.
- Snell, J., Swersky, K., Zemel, R. (2017). *Prototypical Networks for Few-Shot Learning.* NeurIPS 2017.
- Finn, C., Abbeel, P., Levine, S. (2017). *Model-Agnostic Meta-Learning.* ICML 2017.
- Nichol, A., Achiam, J., Schulman, J. (2018). *On First-Order Meta-Learning Algorithms (Reptile).* arXiv:1803.02999.
- Valiant, L.G. (1984). *A Theory of the Learnable.* CACM 27(11).
- Sauer, N. (1972). *On the Density of Families of Sets.* J. Combinatorial Theory.
- Belkin, M., et al. (2019). *Reconciling Modern ML Practice and the Classical Bias-Variance Trade-Off.* PNAS.
- Bartlett, P.L., et al. (2020). *Benign Overfitting in Linear Regression.* PNAS.
- McAllester, D. (1999). *PAC-Bayesian Model Averaging.* COLT 1999.

E.4 Architetture, Vision, Time Series, Multimodale

- Fedus, W., Zoph, B., Shazeer, N. (2022). *Switch Transformers: Scaling to Trillion Parameter Models.* JMLR.

- Jiang, A.Q., et al. (2024). *Mixtral of Experts.* arXiv:2401.04088.
- Dao, T., et al. (2022). *FlashAttention: Fast and Memory-Efficient Exact Attention with IO-Awareness.* NeurIPS 2022.
- Liu, Z., et al. (2021). *Swin Transformer: Hierarchical Vision Transformer using Shifted Windows.* ICCV 2021.
- Gilmer, J., et al. (2017). *Neural Message Passing for Quantum Chemistry.* ICML 2017.
- Xu, K., et al. (2019). *How Powerful Are Graph Neural Networks?* ICLR 2019.
- Kirkpatrick, J., et al. (2017). *Overcoming Catastrophic Forgetting in Neural Networks.* PNAS.
- Lopez-Paz, D., Ranzato, M. (2017). *Gradient Episodic Memory for Continual Learning.* NeurIPS 2017.
- Buzzega, P., et al. (2020). *Dark Experience for General Continual Learning (DER++).* NeurIPS 2020.
- Cohen, J., Rosenfeld, E., Kolter, J.Z. (2019). *Certified Adversarial Robustness via Randomized Smoothing.* ICML 2019.
- Madry, A., et al. (2018). *Towards Deep Learning Models Resistant to Adversarial Attacks.* ICLR 2018.
- Zhang, H., et al. (2019). *TRADES: Theoretically Principled Trade-off between Robustness and Accuracy.* ICML 2019.
- Guo, C., et al. (2017). *On Calibration of Modern Neural Networks.* ICML 2017.
- Lakshminarayanan, B., Pritzel, A., Blundell, C. (2017). *Simple and Scalable Predictive Uncertainty Estimation Using Deep Ensembles.* NeurIPS 2017.
- Liu, W., et al. (2020). *Energy-Based Out-of-Distribution Detection.* NeurIPS 2020.
- Alayrac, J.B., et al. (2022). *Flamingo: A Visual Language Model for Few-Shot Learning.* NeurIPS 2022.
- Li, J., et al. (2023). *BLIP-2: Bootstrapping Language-Image Pre-training with Frozen Image Encoders and LLMs.* ICML 2023.
- Liu, H., et al. (2023). *Visual Instruction Tuning (LLaVA).* NeurIPS 2023.

- Lewis, P., et al. (2020). *Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks.* NeurIPS 2020.
- Carion, N., et al. (2020). *End-to-End Object Detection with Transformers (DETR).* ECCV 2020.
- Kirillov, A., et al. (2023). *Segment Anything (SAM).* ICCV 2023.
- Lim, B., et al. (2021). *Temporal Fusion Transformers for Interpretable Multi-Horizon Time Series Forecasting.* Int. J. Forecasting.
- Oreshkin, B.N., et al. (2020). *N-BEATS: Neural Basis Expansion Analysis for Interpretable Time Series Forecasting.* ICLR 2020.
- Nie, Y., et al. (2023). *A Time Series is Worth 64 Words: Long-Term Forecasting with Transformers (PatchTST).* ICLR 2023.
- Rajbhandari, S., et al. (2020). *ZeRO: Memory Optimizations Toward Training Trillion Parameter Models.* SC 2020.
- Frantar, E., et al. (2023). *GPTQ: Accurate Post-Training Quantization for Generative Pre-trained Transformers.* ICLR 2023.
- Shazeer, N., Stern, M. (2018). *Adafactor: Adaptive Learning Rates with Sublinear Memory Cost.* ICML 2018.
- Chen, X., et al. (2023). *Symbolic Discovery of Optimization Algorithms (Lion).* NeurIPS 2023.